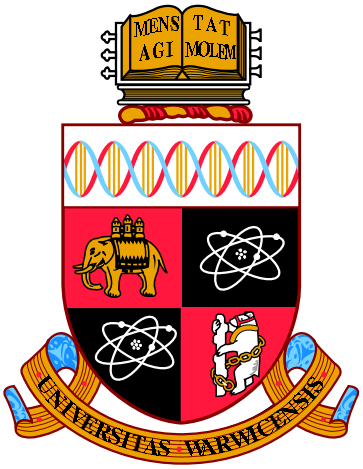


On a Quasi-Stationary Approach to Bayesian Computation, with Application to Tall Data

by

Divakar Kumar

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Statistics

September 2018

To Govind baba and my family

Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my sincere gratitude to my supervisors Prof. Gareth Roberts and Dr Murray Pollock for their insightful guidance and motivation. Their continued support and encouragement had been instrumental throughout the entirety of my PhD. I could not have imagined having better advisers for my research studies.

I would extend my sincere gratitude to Prof. Omiros Papaspiliopoulos (Universitat Pompeu Fabra) for his assistance and showing a new direction for my research project. I owe my special thanks to my colleague Andi Wang (University of Oxford) for a number of constructive discussions on my research work.

I would be failing in my duty if I do not express my heartfelt thanks to my friends whose support and encouragement was worth more than I can express on paper. This is a long list but some key people deserve a special mention: Kinshuk S. Rattu (University College London), Cyril Chimisov (Google Inc.), Sophia K. Wright (University of Warwick), Sanskriti Jain (Delhi University), Shyam Popat (University of Warwick).

This journey would not have been possible without the support and unparalleled love of my family members. I am forever indebted to my parents for inspiring me to pursue a PhD, encouraging me to explore new avenues in life and to seek my own destiny. I dedicate this work to my mother and late father who have sacrificed their comforts for my studies.

Last but not the least, I would like to acknowledge the Oxford-Warwick Statistics Program (OxWaSP), Department of Statistics, University of Warwick and the Engineering and Physical Sciences Research Council (EPSRC) for showing interest in my research abilities and the financial support.

Divakar Kumar

September 30, 2018

Declarations

I declare that the research presented in this thesis is the result of my own work, unless otherwise stated. I confirm that my work has been prepared in accordance with the guidelines set by the University of Warwick on the presentation of a research thesis. This thesis has been submitted to the University of Warwick only and not to any other institution, towards my examination for the Doctor of Philosophy.

Signed: 

Divakar Kumar

September 30, 2018

Abstract

Markov Chain Monte Carlo (MCMC) techniques have traditionally been used in a Bayesian inference to simulate from an intractable distribution of parameters. However, the current age of Big data demands more scalable and robust algorithms for the inferences to be computationally feasible. Existing MCMC-based scalable methodologies often uses discretization within their construction and hence they are inexact. A newly proposed field of the Quasi-Stationary Monte Carlo (QSMC) methodology has paved the way for a scalable Bayesian inference in a Big data setting, at the same time, its exactness remains intact. Contrary to MCMC, a QSMC method constructs a Markov process whose quasi-stationary distribution is given by the target. A recently proposed QSMC method called the Scalable Langevin Exact (ScaLE) algorithm has been constructed by suitably combining the exact method of diffusion, the Sequential Monte Carlo methodology for quasi-stationarity and sub-sampling ideas to produce a sub-linear cost in a Big data setting. This thesis uses the mathematical foundations of the ScaLE methodology as a building block and carefully combines a recently proposed regenerative mechanism for quasi-stationarity to produce a new class of QSMC algorithm called the Regenerating ScaLE (ReScaLE). Further, it provides various empirical results towards the sub-linear scalability of ReScaLE and illustrates its application to a real world big data problem where a traditional MCMC method is likely to suffer from a huge computational cost. This work takes further inroads into some current limitations faced by ReScaLE and proposes various algorithmic modifications for targeting quasi-stationarity. The empirical evidences suggests that these modifications reduce the computational cost and improve the speed of convergence.

Abbreviations

B-vM Bernstein von-Mises pp. vi, 19, 90, 99

ESS Effective Sample Size pp. vi, 19, 20, 85, 89, 91

MCMC Markov Chain Monte Carlo pp. xv, 2–4, 76, 108, 112, 140, 142

QSD Quasi Stationary Density pp. 48, 132

QSMC Quasi-Stationary Monte Carlo pp. iii, xv, 4, 6, 60, 64, 76, 106, 108, 140, 142, 170

ReScaLE Regenerating Scalable Langevin Exact pp. iv, vi, viii–xi, xv, 6, 7, 76, 78, 79, 82–85, 87, 89–91, 94, 97–99, 98, 99, 101, 102, 104, 106, 108, 106, 108, 112, 114, 118, 119, 122, 119, 122, 124, 122, 124, 127, 129, 130, 132–135, 134, 135, 138, 140, 142, 162, 169

RWM Random Walk Metropolis pp. vi, 90, 99, 101, 102

ScaLE Scalable Langevin Exact pp. iii, xv, 4, 7, 60, 69, 72, 73, 76, 97, 108, 140, 142

SGLD Stochastic Gradient Langevin Dynamics pp. 3, 4

SGLD-CV Stochastic Gradient Langevin Dynamics with Control Variate p. 4

SMC Sequential Monte Carlo pp. 4, 6, 53, 58, 67, 72, 140

UND Uniform Norm Distance pp. vi, x, xi, 18, 19, 85, 89, 91, 99, 114, 133, 134

US United States pp. iv, x, 76, 97, 98, 112, 122, 124, 140

Chapter 1

Introduction

The explosive growth in new technology has led to a tremendous increase in the size of data being generated, especially the scientific and engineering domains where terabytes and petabytes are becoming increasingly common [Katal et al., 2013; Labrinidis and Jagadish, 2012]. Such complex data streams require a cheap storage facility, greater processing speed and prompt decision making in a highly uncertain environment [Jagadish et al., 2014]. This has produced a bigger challenge in many areas of statistics where the algorithmic cost of the parametric inference does not scale well with data size [Bardenet et al., 2017; Pollock et al., 2017]. Particularly, the domain of Bayesian inference where a user is interested in combining prior information with uncertain evidence, which in turn requires scalable approaches to make a sophisticated parametric inference.

This chapter motivates some computational challenges inherent within a Bayesian parametric inference when a big data set is under consideration. In our context, a big dataset consists of a large number n of individual data points relative to the dimension of each observations, also called *tall data*. In this chapter, we briefly present some scalable approaches to a Bayesian inference on a tall data problem, together with their strengths and limitations. This chapter is organised as follows: Section (1.1) lays down the motivation behind this work coupled with existing approaches to performing parametric inference on tall data. Section (1.2) outlines a brief summary of this work, of which the structure is analysed in Section (1.3).

1.1 Motivation : Big Data is a Problem

One motivation for the methodology developed in this work stems from the following situation: suppose we are given $n \gg 1$ observations and we are interested in the Bayesian inference for the parameter $\mathbf{x} \in \mathbb{R}^d$. We are interested in obtaining samples according to the following posterior density:

$$\pi(\mathbf{x}) \propto f_0(\mathbf{x}) \prod_{i=1}^{n} f_i(\mathbf{x}), \tag{1.1.1}$$

where $f_0(\mathbf{x})$ is a prior and $\{f_i(\mathbf{x})\}_{i=1}^n$ are likelihood factors. In a Bayesian setting, a user could employ the Markov Chain Monte Carlo (MCMC) techniques in order to obtain samples from $\pi(\cdot)$, where the idea is to construct a Markov process whose invariant density is given by $\pi(\cdot)$ [Gelman et al., 2011; Brooks et al., 2011]. However, standard MCMC algorithms (for example Metropolis-Hastings) are not suited in this case as they require the likelihood evaluation of each datum in each iteration. The evaluation of (1.1.1) at each iteration of an MCMC algorithm is costly as it would require $\mathcal{O}(n)$[1] calculations, which makes such methods highly infeasible from a computational point of view [Bierkens et al., 2016; Pollock et al., 2017].

Another approach for performing inferences on big data is *Consensus Monte Carlo* [Scott et al., 2016; Bardenet et al., 2017]. The Consensus Monte Carlo approach has been studied frequently in literature, with [Wang et al., 2016b; Huang and Gelman, 2005; Scott, 2017; Neiswanger et al., 2013] to name a few. Such methods are employed when the data is too big to fit into a memory at once, therefore, the data is *divided* into $N_{cluster}$ pieces and saved on separate clusters. The Bayesian inferences are then performed in parallel on each cluster separately and the resulting estimates are 'recombined'. Although the divide-and-conquer approach reduces the computational cost by a factor of $N_{cluster}$, the recombination step is often inexact and approximate which tends to work only when the posterior takes the Gaussian form.

To obtain Monte Carlo draws according to π, another approach that a user could employ is to exploit the fact that the *Langevin diffusion* has a known invariant distribution. The Langevin diffusion corresponding to the target density π is given

[1] Recall that a function $f(x) = \mathcal{O}(g(x))$ if and only if there exists a $C > 0$ and a $x_0 \in \mathbb{R}$ such that $|f(x)| \leq C|g(x)|$ for all $x \geq x_0$.

by

$$d\mathbf{X}_t = \frac{1}{2}\nabla \log(\pi(\mathbf{X}_t))dt + d\mathbf{W}_t, \quad \mathbf{X}_0 = \mathbf{x} \in \mathbb{R}^d, t \in [0, \infty), \tag{1.1.2}$$

where $\mathbf{W}_t$ is a $d-$dimensional Brownian motion. The invariant density of the diffusion in (1.1.2) is given by π. So the problem of obtaining draws from π boils down to simulating the trajectories of the diffusion in (1.1.2). One approach to simulate a trajectory of (1.1.2) is the *time-discretization* method such as Euler-Maruyama. In the basic implementation of this scheme, we assume that the position of a trajectory at time $t + h$ can be approximated by a Gaussian density namely

$$\mathbf{X}_{t+h}|(\mathbf{X}_t = \mathbf{x}_t) = \mathbf{x}_t + h\frac{1}{2}\nabla \log(\pi(\mathbf{x}_t)) + \epsilon_t \text{ where } \epsilon_t \sim \mathcal{N}(0, h). \tag{1.1.3}$$

However, this approximation carries two fold limitations. First, it requires the evaluation of the drift component in (1.1.2) which would require an $\mathcal{O}(n)$ calculation. Secondly, the size of error due to the discretization bias grows with the widening time-horizon. Controlling the discretization error is computationally expensive which often involves making time-discretization sufficiently negligible [Kloeden and Platen, 1999]. Moreover, such approximations are not exact due to discretization and it does not converge to the correct target density [Pollock et al., 2017].

1.1.1 Some Scalable Approaches

Motivated by the nature of the drift term within the Langevin dynamics, one can reduce the $\mathcal{O}(n)$ computational cost in updating (1.1.3) by employing sub-sampling methods which uses only a random subset of data in each update. Such methods are often described under the umbrella of *Stochastic Gradient Langevin Dynamics* (SGLD) [Welling and Teh, 2011; Sato and Nakagawa, 2014]. The SGLD achieves scalability in a large-scale MCMC by replacing the exact gradient in (1.1.3) by a stochastic gradient which uses a small mini-batch of size $m \ll n$. A stochastic gradient is constructed such that it is an unbiased estimator of the full gradient and it is computationally cheaper to evaluate. In particular, one can draw a random subset A of size m from $\{0, ..., n\}$ (in particular, the elements of A are drawn uniformly with replacement from the set $\{0, ..., n\}$) and perform the following update:

$$\mathbf{X}_{t+h}|(\mathbf{X}_t = \mathbf{x}_t) = \mathbf{x}_t + \frac{h}{2}\frac{(n+1)}{m}\sum_{i \in A}\nabla \log(f_i(\mathbf{x}_t)) + \epsilon_t \text{ where } \epsilon_t \sim \mathcal{N}(0, h), \tag{1.1.4}$$

where the unbiased estimator satisfies

$$\mathbb{E}\left(\frac{(n+1)}{m}\sum_{i\in A}\nabla\log(f_i(\mathbf{x}_t))\right)=\sum_{i=0}^{n}\nabla\log(f_i(\mathbf{x}_t))=\nabla\log(\pi(\mathbf{x}_t)). \tag{1.1.5}$$

Another variant of the SGLD employs the construction of a control variate within the sub-sampling step in SGLD, often called Stochastic Gradient Langevin Dynamics with Control Variate (SGLD-CV) [Baker et al., 2017; Nagapetyan et al., 2017]. A control variate is employed to reduce the Monte Carlo variance in the construction of the unbiased estimator in (1.1.4). It is a fixed parametric value $\hat{\mathbf{x}}$ chosen 'close' to the posterior mode. We should note that a SGLD-CV updates as

$$\mathbf{X}_{t+h}|(\mathbf{X}_t=\mathbf{x}_t)=\mathbf{x}_t+\frac{h}{2}\left(\frac{(n+1)}{m}\sum_{i\in A}[\nabla\log(f_i(\mathbf{x}_t))-\nabla\log(f_i(\hat{\mathbf{x}}))]+\nabla\log(\pi(\hat{\mathbf{x}}))\right)+\epsilon_t. \tag{1.1.6}$$

If $\mathbf{x}_t\approx\hat{\mathbf{x}}$ then $[\nabla\log(f_i(\mathbf{x}_t))-\nabla\log(f_i(\hat{\mathbf{x}}))]\approx 0$ for a smooth behaviour of the likelihood function f_i. The variance of the unbiased estimator in SGLD is smaller than SGLD-CV, this is mainly because $\nabla\log(f_i(\mathbf{x}_t))$ and $\nabla\log(f_i(\hat{\mathbf{x}}))$ are highly correlated whenever $[\nabla\log(f_i(\mathbf{x}_t))-\nabla\log(f_i(\hat{\mathbf{x}}))]\approx 0$ [Baker et al., 2017]. Therefore SGLD-CV can be more efficient than its SGLD counterpart. However, both SGLD and SGLD-CV are still inexact due to the discretization step involved and it requires the step size h to be negligibly small in order to gain better convergence. Moreover, SGLD has overall computational cost of $\mathcal{O}(n)$ in spite of reduced computational cost per iteration [Nagapetyan et al., 2017; Welling and Teh, 2011].

Another class of Monte Carlo algorithm which finds its use in the context of big data is *Fire Fly Monte Carlo (FlyMC)* method [Maclaurin and Adams, 2014]. This method works by an introduction of binary auxiliary variables $\{z_i\}_{i=1}^n$ in such a way that they do not change the marginal distribution of the parameter of interest. Informally, such auxiliary variables turn the data points 'on' and 'off' in the posterior distribution of interest. The FlyMC method requires that each likelihood term $f_i(\mathbf{x})$ can be lower bounded by $B_i(\mathbf{x})>0$ such that the joint distribution of the parameter and the auxiliary variables can be written as

$$\pi(\mathbf{x},\{z_i\}_{i=1}^n)\propto\overbrace{\left(f_0(\mathbf{x})\prod_{i=1}^{n}B_i(\mathbf{x})\right)}^{:=\pi_1(\mathbf{x})}\times\overbrace{\left(\prod_{i:z_i=1}\frac{f_i(\mathbf{x})-B_i(\mathbf{x})}{B_i(\mathbf{x})}\right)}^{:=\pi_2(\mathbf{x})}. \tag{1.1.7}$$

We can generate samples according to our target density by alternating between the simulation of $\mathbf{x}$ conditional on $\{z_i\}_{i=1}^n$ according to $\pi_1(\mathbf{x})$ and updates of $\{z_i\}_{i=1}^n$ conditional on $\mathbf{x}$ according to $\pi_2(\mathbf{x})$. We should note that those data points only appear in iterations for which $z_i = 1$, which means that only a mini-batch of data is used while simulating according to $\pi_2(\mathbf{x})$, providing the scalability at each iteration. Here, $B_i(\mathbf{x})$ is conveniently chosen such that $\pi_1(\mathbf{x})$ can be calculated at each iteration with an $\mathcal{O}(1)$ computational cost. This provides the overall scaling property to this method with respect to the data size. However, we should note that a useful lower bound for each likelihood terms can be challenging to obtain. Furthermore, the introduction of the auxiliary variables increases the state-space of this problem drastically which usually results in slower mixing of the MCMC chain [Maclaurin and Adams, 2014].

A recent development utilising an exact approach is the *Zig-Zag* method, whose dynamics depend on the target through the gradient of the target density [Bierkens et al., 2016] [Bouchard-Côté et al., 2018]. In a Bayesian simulation, one can unbiasedly estimate the gradient using the sub-sampling method, while retaining the exactness of the underlying method. Moreover, like SGLD-CV, the Zig-Zag method employs the control variate ideas to reduce the variance of the unbiased estimators of the gradient, which ultimately provides scaling properties to the Zig-Zag method [Pollock et al., 2017]. A Zig-Zag process works by constructing a $d-$dimensional continuous-time Markov process $\{\mathbf{X}_t \,|\, t \geq 0\}$, which moves with constant velocity $\theta(t) \in \{+1, -1\}^d$ until one of its component trajectories (say i^{th}) switches sign at a time-dependent switching rate $\lambda_i(\mathbf{X}_t, \theta(t))$. A Zig-Zag process with rate vector chosen as $\lambda_i(\mathbf{X}_t, \theta(t)) = \max\{\theta_i \nabla_i \log(\pi(\mathbf{X}_t)), 0\}$ retrieves the correct intractable density given by the marginal of the invariant density [Bierkens et al., 2016]. We should note that the form of λ_i allows us to perform sub-sampling which leads to the scalability with respect to the data size. However, the invocation of sub-sampling with control variate changes the underlying algorithm. Furthermore, we should note that the introduction of the auxiliary velocity parameters increases the dimensionality of the underlying problem and hence will result in an increased computational cost.

In order to achieve the scalability of the method in a big data setting while maintaining its exactness at the same time, Pollock et al. [2017] proposed a class of the *Quasi-Stationary Monte Carlo* (QSMC) methods. Contrary to an MCMC method which relies upon the stationarity of a Markov chain, QSMC targets the *quasi-*

stationary distribution of a suitably constructed stochastic process which exhibits a killing mechanism (ref to Chapter (5) for a detailed discussion). More formally, for a Markov process $\{\mathbf{X}_t \,|\, t \geq 0\}$ which is killed using a suitably chosen instantaneous rate $\kappa(\cdot) > 0$ at a random time $\zeta > 0$, the quasi-stationary distribution is given by the limiting behaviour of $\mathbf{X}_t \,|\, \zeta > t$ as $t \to \infty$. Within a QSMC framework, this limiting distribution is chosen as a proxy for the target distribution of interest. Pollock et al. [2017] further uses the control variate method within a subsampling framework to produce the *Scalable Langevin Exact (ScaLE)* algorithm, which is not only exact but also produces a sub-linear cost with respect to the data size. However, the ScaLE algorithm relies on the *Sequential Monte Carlo (SMC)* for its implementation, which in turn requires a large number of *particles* for its convergence [Del Moral et al., 2006; Doucet and Johansen, 2011]. Furthermore, letting the diffusion time $t \to \infty$ further deepens the computational issues within a SMC framework, consequently, the computational object in such a setting can be a big challenge to handle [Pollock et al., 2017].

1.2 Novel Material Summary

This thesis introduces another QSMC algorithm which can be used to sample from a given intractable distribution of interest. It uses the mathematical foundations developed in the recent literature on the QSMC theory [Pollock et al., 2017]. However, unlike [Pollock et al., 2017] we consider a different approach in order to simulate according to the quasi-stationary density of the underlying Markov process. This approach uses an extension of the methodology developed in the works of [Blanchet et al., 2016] which in turn employs a regenerative mechanism to simulate from the quasi-stationary density of interest; thereby naming our resulting QSMC algorithm by *Regenerating ScaLE (ReScaLE)*. In contrast to an SMC approach, the regenerative mechanism works by simply considering one particle which follows the dynamics of a given Markov process until absorption. Upon absorption, it regenerates according to the empirical density of the states visited by the process. This gives us an edge over an SMC sampler as the computational cost incurred in handling one particle is much less.

This thesis further extends the regenerative mechanism developed in [Blanchet et al., 2016] by proposing various rebirth strategies which are then embedded within our QSMC algorithm. We provide various empirical evidences to support the use of the proposed rebirth strategies within our QSMC algorithm. These rebirth strategies

do not only address the current limitations faced by our QSMC algorithm but at the same time, they propose possible extensions to the current literature on the study of quasi-stationary behaviour [Blanchet et al., 2016]. However, this thesis does not study the mathematical side of our QSMC algorithm and the proposed rebirth strategies.

1.3 Thesis Structure

This thesis is broken into two parts: Part I reviews relevant literature which has been used to design the novel material developed in this thesis; whereas Part II presents the development of our methodology together with some applications and its possible extensions.

Part I consists of Chapters 2,3,4 and 5 where we outline a number of existing relevant piece of literature. In Chapter 2 we review a number of elementary sampling algorithms including *rejection sampling, Poisson processes* and *series sampling*. In Chapter 3 we present Brownian motion and some related processes together with their simulation algorithms. We present the *localisation method* – an alternative mechanism of simulating the trajectories of a Brownian motion, which is a key ingredient in the construction of the algorithm developed in this work [Pollock, 2013; Chen and Huang, 2013]. We also outline some key definitions concerning diffusion processes which are later used in this work. Chapter 4 is an overview of the quasi-stationary behaviour of a process and goes on to outline some simulation algorithms in order to simulate according to the quasi-stationary density of a process which exhibits a killing mechanism [Blanchet et al., 2016]. Chapter 5 reviews the most relevant literature consisting of the Quasi-Stationary Monte Carlo methodology, followed by the algorithmic construction of the ScaLE method [Pollock et al., 2017].

Part II consists of Chapters 6 and 7. In particular, Chapter 6 outlines the mathematical and the algorithmic construction of the ReScaLE method. It combines the mathematical frameworks developed in Chapter 5 together with the algorithmic construction of the quasi-stationary density from Chapter 4 and localised Brownian motion outlined in Chapter 3. Furthermore, we consider some applications of the ReScaLE method developed on synthetic and real datasets. Chapter 7 lays out a few algorithmic bottlenecks developed in Chapter 6 together with some extensions in order to circumvent these issues.

Part I

Literature Review

Chapter 2

Some Useful Monte Carlo Methods

In this chapter, we review some Monte Carlo techniques which are instrumental in the development of the methodology in this thesis. We discuss some sampling techqniues such as rejection sampling, importance sampling and Poisson thinning. A detailed account of these methods can be found in [Devroye, 1986; Ross, 2010; Robert and Casella, 2013; Kingman, 1993; Cox and Isham, 1980].

This chapter is organised as follows: Section (2.1) and (2.2) outlines rejection and importance sampling techniques respectively. We present a brief discussion of Poisson processes together with its simulation algorithms in Section (2.3). Section (2.4) outlines a series sampling method – a method to sample a random variable which has a density which can be expressed as an infinite series. Section (2.5) outlines various performance measures which can be employed within a Monte Carlo framework in order to gauge the quality of samples obtained. These performance metrics are later used throughout the course of this work.

2.1 Rejection Sampling

A computational statistician is often interested in sampling from a density π, samples from which cannot be obtained directly. However, π can be evaluated point-wise up to a constant. In such a setting, a class of Monte Carlo methods called the *Rejection sampling* [von Neumann, 1951; Robert and Casella, 2013] can be used to obtain samples from such distributions. In a rejection sampling setting, one chooses a density q such that it is easy to obtain samples from it. A choice of q is made

such that π is absolutely continuous with respect to q with bounded *Radon-Nikodým* derivative. More formally, with slight abuse of notation we have, [Devroye, 1986, Chapter 2]

$$\sup_{x \in \mathbb{R}} \frac{d\pi}{dq}(x) \leq c < \infty, \tag{2.1.1}$$

for some $0 < c \in \mathbb{R}$. Therefore, one can obtain samples as presented in Algorithm (2.1.1).

Algorithm 2.1.1: REJECTION SAMPLER(π, q, N)

---

output $(\mathcal{S} = \{x_1, ..., x_N\})$
$\mathcal{S} \leftarrow \phi$
for $i \in \{1, ..., N\}$
do
- accepted $\leftarrow$ **false**
- **while** accepted $\neq$ **true**
  - **do**
    - $x_i \sim q, u \sim U[0, 1]$
    - **if** $u \leq \pi(x_i)/(c \times q(x_i))$
      - **do**
        - $\mathcal{S} \leftarrow \mathcal{S} \cup x_i$
        - accepted $\leftarrow$ **true**

return $(\mathcal{S})$

Next we provide an intuitive reasoning behind the working of a rejection sampler; a formal result can be found in [Devroye, 1986, Theorem 3.1]. For simplicity we assume that the univariate random variable X is distributed according to the density π. Simulation of N random points $\{x_1, ..., x_N\}$ according to a density π can be regarded as the simulation of N bivariate points of the form $\{(x_1, y_1), ..., (x_N, y_N)\}$ under the graph of π, where we retain only the first co-ordinates. As it is difficult to simulate points under the graph of π (due to its inaccessibility), we choose another density function q such that N random points of the form $\{(x_i, y_i)\}$ can be easily simulated under the graph of q. A point x_i is retained as a candidate sample for the density π if y_i lies in the interval $(0, c \times q(x_i))$. We show that the conditional distribution of $\{X \leq x\}$ given that $\left\{U \leq \frac{\pi(X)}{(c \times q(X))}\right\}$ yields the distribution function

of π, say F_π. Formally we have

$$\mathbb{P}\left(X \leq x \,\middle|\, U \leq \frac{\pi(X)}{(c \times q(X))}\right) = \frac{\int_{-\infty}^{x} \int_{0}^{\frac{\pi(z)}{(c\times q(z))}} q(z)dudz}{\int_{-\infty}^{\infty} \int_{0}^{\frac{\pi(z)}{(c\times q(z))}} q(z)dudz} \tag{2.1.2}$$

$$= \frac{\int_{-\infty}^{x} \frac{\pi(z)}{(c\times q(z))} q(z)dz}{\int_{-\infty}^{\infty} \frac{\pi(z)}{(c\times q(z))} q(z)dz} \tag{2.1.3}$$

$$= F_\pi(x). \tag{2.1.4}$$

We should further note that the probability of acceptance of a proposed sample X is given by

$$\mathbb{E}_q\left(\frac{\pi(X)}{(c \times q(X))}\right) = \int \frac{\pi(x)}{(c \times q(x))} q(x)dx = \frac{1}{c}. \tag{2.1.5}$$

The number of draws from q needed to obtain a candidate sample for the target π follows a geometric distribution with parameter $1/c$ and hence, smaller values of c would make a rejection sampler more efficient.

2.2 Importance sampling

A rejection sampling can become inefficient when a large number of proposed samples are rejected, this typically occurs when the probability of acceptance is too low. In such situations, it is instrumental to obtain some samples from the interesting regions of the target density. A natural approach to weigh the important regions of the target density is known as importance sampling [Kahn, 1949; Goertzel et al., 1950]. An importance sampler proposes a sample $X \in \mathbb{R}$ according to q, for the target density π and associates a weight of $w(X) := \pi(X)/q(X)$. Suppose we are interested in estimating the quantity $\mathbb{E}_\pi(h(X))$ then we have,

$$\mathbb{E}_\pi(h(X)) = \int_{-\infty}^{\infty} h(x)\pi(x)dx = \int_{-\infty}^{\infty} h(x)\frac{\pi(x)}{q(x)}q(x)dx = \mathbb{E}_q\left(h(X)w(X)\right). \tag{2.2.1}$$

Therefore, one can construct a Monte Carlo estimate of $\mathbb{E}_\pi(h(X))$ by observing that

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} h(X_i)w(X_i) = \mathbb{E}_\pi(h(X)), \text{ where } X_i \sim q. \tag{2.2.2}$$

The limit (2.2.2) holds as a result of the strong law of large numbers [Ross, 2012, Chapter 2.7].

An importance sampler can also be used in a situation when the target density is known upto a normalizing constant, say Z, i.e $\pi = \gamma/Z$. In this setting we denote an unnormalised weight by $w^*(X) = \gamma(X)/q(X)$ and construct an estimator of $\mathbb{E}_\pi(h(X))$ by observing that

$$\frac{\mathbb{E}_q\left(h(X)w^*(X)\right)}{\mathbb{E}_q\left(w^*(X)\right)} = \frac{\int_{-\infty}^{\infty} h(x)\frac{Z\pi(x)}{q(x)}q(x)dx}{\int_{-\infty}^{\infty}\frac{Z\pi(x)}{q(x)}q(x)dx} = \frac{\int_{-\infty}^{\infty} h(x)\pi(x)dx}{\int_{-\infty}^{\infty}\pi(x)dx} = \mathbb{E}_\pi(h(X)). \tag{2.2.3}$$

When we have drawn samples $X_1, ..., X_N$ according to q, we denote the normalized weights by $w(X_i) = w^*(X_i)/\sum_{j=1}^{N} w^*(X_j)$ and our estimator satisfies:

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} h(X_i)\left(w^*(X_i)/\sum_{j=1}^{N} w^*(X_j)\right) = \mathbb{E}_\pi(h(X)). \tag{2.2.4}$$

The above limit (2.2.4) follows as a result of Slutsky's lemma [van der Vaart, 1998, Chapter 2]. This variant of an importance sampler is central to the Sequential Monte Carlo methodology which will be discussed later in Chapter (4). Therefore, we present the algorithmic construction in Algorithm (2.2.1).

Algorithm 2.2.1: IMPORTANCE SAMPLER(γ, q, N)

---

output $(\mathcal{S} = \{(x_1, w^*(x_1)), ..., (x_N, w^*(x_N))\})$
$\mathcal{S} \leftarrow \phi$
for $i \in \{1, ..., N\}$
do $\begin{cases} x_i \sim q \text{ and } w_i^* \leftarrow \gamma(x_i)/q(x_i) \\ \mathcal{S} \leftarrow \mathcal{S} \cup (x_i, w_i^*) \end{cases}$
return $(\mathcal{S})$

2.3 Simulating Poisson Processes

This section outlines Monte Carlo methods for simulating different kinds of Poisson processes. As we will encounter later in this thesis, Poisson processes are a key ingredient in the construction of the methodology discussed. Poisson processes can be used in the construction of a skeleton of a diffusion process while preserving its exactness [Beskos and Roberts, 2005; Beskos et al., 2008]. Therefore, we discuss the simulation method to obtain the event times of a Poisson process. First, we discuss the method for a homogeneous Poisson process which is followed by the inhomogeneous case. We begin with the formal definition:

Definition 2.3.1 ***Poisson Process:*** *A continuous-time stochastic process* $\{N(t)\}_{t\geq 0} \in \mathbb{N} \cup \{0\}$ *with rate* $\lambda(t)$ *is called a Poisson process if it satisfies the following conditions:*

1. *The number of events occurring at time* 0 *i.e.* $N(0) = 0$,
2. *For some* $0 < t < t+s$, *the number of events occurring in the interval* $[t, t+s]$ *has a Poisson distribution. More formally,*

$$\mathcal{P}\left(N(t+s) - N(t) = k\right) = \exp\left(-\int_t^{t+s} \lambda(x)dx\right) \frac{\left(\int_t^{t+s} \lambda(x)dx\right)^k}{k!}. \tag{2.3.1}$$

3. *The number of events occurring in any two disjoint time intervals* $[r, r+s]$ *and* $[t, t+u]$ *are independent i.e.* $N(t+u) - N(t) \perp N(r+s) - N(r)$.

2.3.1 Simulating A Homogeneous Poisson Process

Next, we will outline the simulation mechanism of the event times $0 = t_0 < t_1 < t_2 ...$ of a Poisson process, when the rate parameter is constant i.e. $\lambda(t) = \lambda$. This is often termed as a homogeneous Poisson process. To achieve this, we can use the fact that the waiting time between consecutive events is exponentially distributed [Kingman, 1993; Ross, 2010]. If $T_i := t_i - t_{i-1}$ denotes the i^{th} waiting time (with $T_0 = 0$) and each T_i is exponentially distributed, then the algorithm to simulate the event times in a given interval $[0, t]$ can be summarised as given in Algorithm (2.3.1).

Algorithm 2.3.1: HOMOGENEOUS POISSON PROCESS SIMULATOR(λ, t)

output $(t_1, ..., t_{k-1})$
$i \leftarrow 0; t_i \leftarrow 0$
while $t_i \leq t$
do $\begin{cases} i \leftarrow i + 1 \\ T \sim Exp(\lambda) \\ t_i \leftarrow t_{i-1} + T \end{cases}$
$k \leftarrow \min\{i : t_i > t\}$
return $(t_1, ..., t_{k-1})$

2.3.2 Simulating Inhomogeneous Poisson Process Using Thinning

In this section we outline the methodology to simulate the event times of a time-inhomogeneous Poisson process of rate $\lambda(t)$. We achieve this by employing a dominating homogeneous Poisson process with a constant rate Λ chosen such that $\lambda(t) \leq \Lambda$ for all t. We further use *Poisson thinning* (also called colouring (see [Kingman, 1993])) to perform simulations for our inhomogeneous target. We first observe that for a time-homogeneous Poisson process $\{N(t) : t \geq 0\}$ with arrival rate Λ where each event is coloured either red or green with probability p and $1 - p$ respectively, the number of events coloured as red given the total number of events $N(t)$ follow a binomial distribution. Formally, we denote by $N_1(t)$ and $N_2(t)$ the number of events coloured as red and green respectively. Observing that $N(t) = N_1(t) + N_2(t)$, we have

$$\mathbb{P}(N_1(t) = i, N_2(t) = j) = \sum_{k=0}^{\infty} \mathbb{P}(N_1(t) = i, N_2(t) = j \,|\, N(t) = k)\mathbb{P}(N(t) = k) \tag{2.3.2}$$

$$= \mathbb{P}(N_1(t) = i, N_2(t) = j \,|\, N(t) = i + j)\mathbb{P}(N(t) = i + j) \tag{2.3.3}$$

$$= \frac{(i+j)!}{i!\, j!} p^i (1-p)^j \times \exp(-\Lambda t) \frac{(\Lambda t)^{i+j}}{(i+j)!} \tag{2.3.4}$$

$$= \exp(-\Lambda p t) \frac{(\Lambda p t)^i}{i!} \times \exp(-\Lambda(1-p)t) \frac{(\Lambda(1-p)t)^j}{j!}. \tag{2.3.5}$$

Based on the above factorization, it is evident that the number of red and green events follows a Poisson distribution with rate Λp and $\Lambda(1-p)$ respectively. Therefore, it can be concluded that the two processes $\{N_1(t) : t \geq 0\}$ and $\{N_2(t) : t \geq 0\}$ are Poisson processes with rate Λp and $\Lambda(1-p)$ respectively.

This result has an immediate consequence in the simulation of a time-inhomogeneous Poisson process of rate $\lambda(t)$. For $\lambda(t) \leq \Lambda$ we can view a time-homogeneous Poisson process with rate Λ as the total number of red and green events, each following a Poisson process with rate $\lambda(t)$ and $\Lambda - \lambda(t)$ respectively. In this setting, an event at time $q \in [0,t]$ arising as a result of a homogeneous Poisson process with rate Λ is coloured red with probability $p = \lambda(q)/\Lambda$. Therefore, a time-inhomogeneous Poisson process can be simulated by first simulating a time-homogeneous Poisson process and then classifying the events with the aforementioned probability. We synthesize the above arguments in the Algorithm (2.3.2).

Algorithm 2.3.2: INHOMOGENEOUS POISSON PROCESS SIMULATOR$(\lambda(t), \Lambda, t)$

---

output $(s_1, ..., s_j)$
$i \leftarrow 0; t_i \leftarrow 0; j \leftarrow 0$
while $t_i \leq t$
do
- $i \leftarrow i + 1$
- $T \sim Exp(\Lambda)$
- $t_i \leftarrow t_{i-1} + T$
- $U \sim U[0,1]$
- **if** $U \leq \lambda(t_i)/\Lambda$
  - **do**
    - $j \leftarrow j + 1$
    - $s_j \leftarrow t_i$

return $(s_1, ..., s_j)$

2.4 Series Sampling

We encounter many instances in this thesis where we are interested in simulating a random variable X having a density f, which can be expressed as an infinite series i.e. $f(\cdot) = \sum_{i=0}^{\infty} a_i(\cdot)$. The exact evaluation of f with a motive of performing rejection sampling (under $f \leq ch$ for some tractable density h, as in Section (2.1))

is infeasible due to the infinite sum. Although one can evaluate the partial sum of f which guarantees high accuracy, this method still remains inexact. Therefore, for the purpose of drawing exact samples according to the density f we assume that f can be approximated from above and below by a sequence of functions f_n and g_n, which can be evaluated easily. More formally we have

1. $\lim_{n\to\infty} f_n = f$ and $\lim_{n\to\infty} g_n = f$ such that $f_n \leq f \leq g_n$ for all n, (2.4.1)
2. $f \leq ch$ for some constant $c > 0$ such that it is easy to simulate from h. (2.4.2)

Under the two conditions above, a series sampler can be implemented which avoids the exact evaluation of f [Devroye, 1986, 1980]. A series sampler can be implemented as follows: we generate $X \sim h$ and iteratively compute the lower $f_n(X)$ and the upper $g_n(X)$ bounding sequences until

- $Uch(X) \leq f_n(X)$ for some $U \sim U[0,1]$ for some n. In this case $Uch(X) \leq f(X)$ and we accept the proposed sample X similar to rejection sampling. Otherwise,
- $Uch(X) \geq g_n(X)$, which implies that $Uch(X) \geq f(X)$. Therefore we reject the proposed sample X.

The methodology to simulate according to f has been synthesized in Algorithm (2.4.1).

Algorithm 2.4.1: SERIES SAMPLER(f, f_n, g_n, c, h)

---

output (X)
accepted $\leftarrow$ **false**
while accepted $\neq$ **true**
do $\begin{cases} n \leftarrow 1; U \sim U[0,1]; X \sim h \\ \textbf{while } f_n(X) \leq Uch(X) \leq g_n(X) \\ \quad \textbf{do } \{n \leftarrow n+1 \\ \textbf{if } Uch(X) \leq f_n(X) \\ \quad \textbf{do } \{\text{accepted} \leftarrow \textbf{true} \end{cases}$
return (X)

A series sampler requires us to have a tractable form of the lower and the upper bounding sequences namely, $f_n(\cdot)$ and $g_n(\cdot)$ respectively. However, this restriction

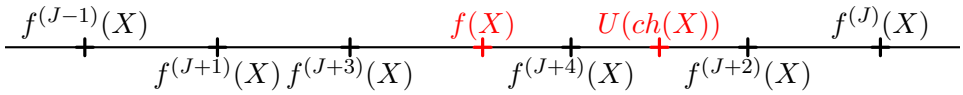


Figure 2.1: *This figure illustrates the oscillating nature of the sequence $f^{(J)}(X)$. It should be noted that the sequence $f^{(J-1)}(X), f^{(J+1)}(X), f^{(J+3)}(X), ...$ converges to $f(X)$ from below while the sequence $f^{(J)}(X), f^{(J+2)}(X), f^{(J+4)}(X), ...$ converges from above. Thus $(U(ch(X)) - f^{(J)}(X))$ and $(U(ch(V)) - f^{(J+1)}(X))$ will have differing signs until the comparison stops.*

can be relaxed in a situation where the the partial sums of f defined by $f^{(J)}(\cdot) := \sum_{j=0}^{J} a_j(\cdot)$ possess the *oscillating property* defined in the following sense:

Definition 2.4.1 (Burq and Jones [2008]) *A sequence of partial sums $\{f^{(J)}(\cdot)\}_{J\geq 1}$ have an oscillating property if there exists an integer N such that for all $J \geq N$, it satisfies*

$$0 \leq -\frac{\left(f^{(J+1)}(\cdot) - f^{(J)}(\cdot)\right)}{\left(f^{(J)}(\cdot) - f^{(J-1)}(\cdot)\right)} \leq 1. \tag{2.4.3}$$

We should note that the sequence of partial sums $\{f^{(J)}(\cdot)\}_{J\geq 1}$ oscillate around the true value of $f(\cdot)$. Without loss of generality we can construct the lower and upper bounding sequences as:

$$f_J(\cdot) := f^{(2J)}(\cdot) \quad \text{and} \quad g_J(\cdot) := f^{(2J+1)}(\cdot) \text{ for } J \geq N; \tag{2.4.4}$$

this is also presented in figure (2.1).

Once we propose a sample $X \sim h$ and generate $U \sim U[0,1]$, we aim to compare $Uch(X)$ to $f_J(X)$ and $g_J(X)$. However, we should note that due to the oscillating nature of the partial sums (and therefore f_J, g_J), the expressions $(Uch(X) - f_J(X))$ and $(Uch(X) - g_J(X))$ bear different signs as long as $f_J(X) \leq Uch(X) \leq g_J(X)$ (see figure (2.1)). $f_J(X) \leq Uch(X) \leq g_J(X)$ is equivalent to $f^{(J)}(X) \leq Uch(X) \leq f^{(J+1)}(X)$. As a result we continue checking the signs of $(Uch(X) - f^{(J)}(X))$ and

$(Uch(X) - f^{(J+1)}(X))$ until they carry the same sign for $J \geq N$. Once this occurs, a decision on the proposed sample can be made as follows:

$$f^{(J)}(X) < f^{(J+1)}(X) < Uch(X) \implies f(X) < Uch(X) \text{ and} \tag{2.4.5}$$
$$f^{(J)}(X) > f^{(J+1)}(X) > Uch(X) \implies f(X) > Uch(X) \text{ for some } J. \tag{2.4.6}$$

We reject the proposed sample X in the former case whilst we accept in the latter case. This procedure can be algorithmically summarized as :

Algorithm 2.4.2: ALTERNATING SERIES SAMPLER$(f^{(\cdot)}, c, h, N)$

---

output (X)
accepted $\leftarrow$ **false**
while accepted $\neq$ **true**
do $\begin{cases} X \sim h; U \sim U[0,1]; J \leftarrow N \\ W \leftarrow U\,(ch(X)) \\ \textbf{while } sign(W - f^{(J)}(X)) \neq sign(W - f^{(J+1)}(X)) \\ \quad \textbf{do } J \leftarrow J + 1 \\ \textbf{if } W \leq f^{(J+1)}(X) \\ \quad \textbf{then } \{\text{accepted} \leftarrow \textbf{true} \end{cases}$
return (X)

For a discussion on the computational cost involved, we refer to [Devroye, 1986, Chapter IV.5].

2.5 Measuring the Quality of Samples

Once we obtain samples using our favourite Monte Carlo method, we are often interested in validating its quality. Therefore, in this section we lay down a number of measures which will be used to analyse the quality of the output obtained using the Monte Carlo methods described in this work.

2.5.1 Uniform Norm Distance and Related Metric

Our first method to measure the closeness of two probability distribution functions F_1 and F_2 is the uniform norm distance (UND) which is defined as

$$d_{un}(F_1, F_2) := \sup_{\mathbf{x}\in\mathbb{R}} |F_1(\mathbf{x}) - F_2(\mathbf{x})|, \tag{2.5.1}$$

When F_1 is unknown, we use its empirical distribution function $\hat{F}_1$ together with finitely many points chosen from the support of F_1, $x_1, ..., x_n$ to estimate

$$\hat{d}_{un}(F_1, F_2) := \max_{x\in\{x_1,...,x_n\}} |\hat{F}_1(x) - F_2(x)| \tag{2.5.2}$$

In a tall data setting, we often use the UND between the empirical distributions resulting due to a given 'method' and the Bernstein-von-Mises (B-vM) approximation i.e. $\hat{d}_{un}(\hat{F}_{method}, \hat{F}_{B-vM})$ [van der Vaart, 1998, Chapter 10.2] as a measure of discrepancy between the two distributions. Here $\hat{F}_{method}$ denotes the empirical distribution function of the chain resulting from the given 'method' and $\hat{F}_{B-vM}$ is the corresponding Bernstein-von-Mises normal approximation. In a context where we compare 'method-1' with respect to 'method-2', we use $\hat{d}_{un}(\hat{F}_{method_1}, \hat{F}_{method_2})$ as a measure of discrepancy between resulting empirical distributions.

In a Bayesian context, we are often interested in measuring the speed of convergence for a given Monte Carlo method to the posterior distribution of interest. While $\hat{d}_{un}$ measures the closeness of two distributions, it does not tell us the speed at which the two distributions become close. Therefore, we discuss another measure of discrepancy defined as the first time the uniform norm distance falls below a given threshold. More formally, for a given threshold $\epsilon > 0$, we define the first time the uniform norm distance is below ϵ as

$$\tau_{und}(\epsilon) := \inf\{t > 0 : \hat{d}_{un}(\hat{F}_{method,t}, \hat{F}_{2,t}) < \epsilon\}, \tag{2.5.3}$$

where $\hat{F}_{method,t}$ (similarly $\hat{F}_{2,t}$) is obtained using a run of the given 'method' upto a maximum diffusion time t.

2.5.2 Effective Sample Size and Efficiency

After having drawn samples from the posterior distribution, we are often interested in knowing how much information samples of a given parameter carry. If there is a strong dependence between the successive draws for a given parameter, we would

expect that our samples do not carry as much information about the posterior distribution compared to draws that are independent. In the Monte Carlo literature, the latency of information is often measured using the *Effective Sample Size (ESS)* [Priestley, 1981; Brooks et al., 2011]. In our case the ESS is defined as:

Definition 2.5.1 ***Effective Sample Size (ESS):*** *Given a sample* $\mathbf{x}_1, ..., \mathbf{x}_N$ *of size* N*, the effective sample size is a measure of the number of independent samples that can be drawn from this set. Mathematically,*

$$ESS := \frac{N}{1 + 2\sum_{k=1}^{\infty} \rho(k)}, \tag{2.5.4}$$

where $\rho(k) = Corr(\mathbf{x}_1, \mathbf{x}_{k+1})$ *[Brooks et al., 2011].*

The denominator term in the definition of ESS, namely, $1 + 2\sum_{k=1}^{\infty} \rho(k)$ is also called the *Integrated Autocorrelation Time (IAT)*. This can be interpreted as the average number of samples required to obtain an independent sample. We use the *batch-mean* method (see [Gilks et al., 1996, Chapter 3.4] for further details) to estimate the effective sample size. This can be achieved by simply using the `ess` function from `mcmcse` package in `R` to estimate the effective sample size [Flegal et al., 2017]. However, we should note that the ESS alone can be misleading as a measure of the mixing achieved by an algorithm. A chain with high ESS (and hence good mixing) can be very slow in terms of its computational cost. At the same time, a chain with low ESS (hence poor mixing property) can be far more computationally efficient. Therefore, a natural measure of performance that combines ESS and the computational cost can be defined in terms of its *efficiency*. [de Valpine et al., 2017]

Definition 2.5.2 ***Efficiency:*** *Efficiency is defined as the effective sample size per unit of computational cost. Formally,*

$$Efficiency := \frac{ESS}{Computational\ cost}. \tag{2.5.5}$$

Efficiency can be interpreted as the number of independent samples that can be drawn per unit of the computational cost. Here we should note that the the use of metrics such as IAT and efficiency has mostly been justified in the Markovian setting when stationarity holds. Therefore, we assume that the stationary holds in the later parts of this work.

2.5.3 The Gelman–Rubin Convergence Diagnostics

In a situation when our target is unknown, we are often posed with a question: has the simulated chain fully explored the support of our target density? Gelman and Rubin [1992] proposed a methodology for assessing the convergence based on multiple chains of a given method. This method works by comparing the between-chains and within-chain variances; when large differences are observed between these variances, it implies non-convergence.

For a formal discussion we assume that we run m different chains of our method, where each chain is observed upto the same length, say t (for simplicity). Let us assume that for i^{th} chain $\hat{x}_i$ and $\hat{\sigma}_i^2$ denote the estimate of our posterior mean and variance respectively. Furthermore, assume that our overall posterior mean for the parameter x is given by $\hat{x} = \frac{1}{m}\sum_{i=1}^{m}\hat{x}_i$. Following the above notations, the between-chain and within-chain variances are given by

$$B := \frac{t}{m-1}\sum_{i=1}^{m}(\hat{x}_i - \hat{x})^2 \text{ and} \tag{2.5.6}$$

$$W := \frac{1}{m}\sum_{i=1}^{m}\hat{\sigma}_i^2 \tag{2.5.7}$$

respectively. We then estimate the variance of our stationary distribution for parameter x as [Gelman and Rubin, 1992]

$$\hat{V}(x) = \left(1 - \frac{1}{t}\right)W + \frac{(m+1)}{mt}B. \tag{2.5.8}$$

Under certain regularity conditions $\hat{V}(x)$ is an unbiased estimator of the posterior variance of our parameter x (see [Gelman and Rubin, 1992; Brooks and Gelman, 1998] for details). We can then define the *potential scale reduction factor (PSRF)* (also called shrink factor or Gelman-Rubin statistic) as [Brooks and Gelman, 1998]

$$\hat{R} := \sqrt{\frac{(d+3)\hat{V}(x)}{(d+1)W}}, \tag{2.5.9}$$

where $d = 2 \times \hat{V}^2/Var(\hat{V})$. $\hat{R}$ estimates the decrease in the in between-chain variability B with respect to the within-chain variability W for all chains under consideration. A value of $\hat{R} < 1.1$ for all model parameters suggests that the convergence has been reached [Brooks and Gelman, 1998]. At the same time, a value

of $\hat{R}$ significantly bigger than 1.1 signifies a lack of convergence to the target density. In this thesis we will use `R` function `coda::gelman.diag()` to estimate PSRF [Plummer et al., 2006].

Chapter 3

Path-Space Simulation of Brownian Motion and Related Processes

This chapter reviews a number of simulation mechanisms pertaining to the trajectories of a *Brownian motion* which are of particular relevance to the methodology developed in this thesis. In particular, the majority of our discussion revolves around the *Layered Brownian motion*, which is a methodology for simulating the trajectories of a Brownian motion sample path under various restrictions.

The current chapter is organised as follows: Section (3.1) defines a simple Brownian motion and outlines an algorithm to obtain sample paths of a univariate Brownian motion. This is followed by a discussion of diffusion processes and the path-space rejection sampling in Section (3.2). We also introduce some key terms which will be used throughout this thesis. In Section (3.3), we introduce the *Localisation method* which is an alternative way to simulate the trajectories of a Brownian motion. The following Sections namely (3.4), (3.5) are key ingredients in the construction of the *Localisation method*. Section (3.4) outlines a method for simulating the *first passage time* and its position for a $d-$ dimensional Brownian motion. Section (3.5) and (3.6) describe a method to simulate the position of a Brownian motion under various restrictions.

3.1 Brownian Motion

The discovery of Brownian motion dates back to 1827 when famous Scottish botanist Robert Brown studied the behaviour of random particles suspended in water. Later in 1905, Einstein gave the mathematical formalism to the concept of a Brownian motion. A detailed mathematical account of Brownian motion can be found in the literature like [Kloeden and Platen, 1999], [Karatzas and Shreve, 1991], [Revuz and Yor, 1999],[Mansuy and Yor, 2008] and [Øksendal, 2003]. To begin with, we can define a Brownian motion as:

Definition 3.1.1 ***Brownian motion:*** *A Brownian motion* $\{W_t \in \mathbb{R} \,|\, t \geq 0\}$ *is a continuous-time stochastic process which satisfies the following properties:*

- $W_0 = 0$.
- *For any* $0 \leq s < t \leq u < v$ *the increments namely* $W_v - W_u$ *and* $W_t - W_s$ *are independent of each other. Furthermore, for any* $0 \leq s < t$, $W_t - W_s \sim N(0, t-s)$.
- *The mapping* $t \mapsto W_t$ *is continuous with probability 1.*

Brownian motion is an infinite-dimensional random variable and hence it is not possible to simulate the entire sample path of its trajectories. However, we can simulate a Brownian motion at any finite number of time points. We can use the properties in the definition (3.1.1) to simulate the positions of a Brownian motion at a given set of time points $t_1, ..., t_n$. This is outlined in the algorithm (3.1.1).

Algorithm 3.1.1: BROWNIAN MOTION SIMULATOR$(t_1, ..., t_n)$

---

output $(W_{t_1}, ..., W_{t_n})$
$t_i \leftarrow 0; W_{t_i} \leftarrow 0$
for $i \in \{1, .., n\}$
  do $\Big\{ W_{t_i} \sim N(W_{t_{i-1}}, t_i - t_{i-1})$
return $(W_{t_1}, ..., W_{t_n})$

We are often interested in the transformations of a Brownian motion which leaves its properties invariant. One such property which is of particular interest is *scaling* [Karlin and Taylor, 1975, Chapter 7.4]. This is given as follows:

Property 3.1.1 ***Scaling Property :*** *For a given Brownian motion* W_t*, the scaled process defined by* $\tilde{W}_t := \frac{1}{c} W_{c^2 t}$ *for some* $c > 0$ *is also a Brownian motion.*

3.1.1 Brownian Bridge

Often we are interested in simulating the position of a Brownian motion at an intermediate point $q \in (s,t)$, given its position W_s and W_t. Intuitively, the process is 'tied down' at both ends, therefore, the intermediate positions of a Brownian motion form a bridge; such a process is often called a *Brownian bridge*. In order to achieve the law of a Brownian bridge process, we first make the following observation on the conditional density:

$$\begin{aligned} p(W_q = z \,|\, W_s = x, W_t = y) &\propto p(W_s = x, W_q = z, W_t = y) \\ &\propto p(W_t = y \,|\, W_s = x, W_q = z) \times p(W_q = z \,|\, W_s = x) \\ &\propto p(W_t = y \,|\, W_q = z) \times p(W_q = z \,|\, W_s = x), \end{aligned} \tag{3.1.1}$$

which yields

$$p(W_q = z \,|\, W_s = x, W_t = y) \propto N\left(x + \frac{(q-s)}{(t-s)}(y-x), \frac{(t-q)(q-s)}{(t-s)}\right). \tag{3.1.2}$$

This result allows us to simulate the intermediate position W_q of a Brownian motion given its positions W_s and W_t. This forms the basis of the simulation of a number of more complex processes which we shall encounter later in this chapter. We present the simulation mechanism at time points $s < q_1 < ... < q_n < t$ given its positions W_s and W_t; this is illustrated in the Algorithm (3.1.2).

Algorithm 3.1.2: BROWNIAN BRIDGE SIMULATOR$(W_s, W_t, q_1, ..., q_n)$

---

output $(W_{q_1}, ..., W_{q_n})$
$q_0 \leftarrow s; W_{q_0} \leftarrow W_s; q_{n+1} \leftarrow t; W_{q_{n+1}} \leftarrow W_t$
for $i \in \{1, .., n\}$
  do $\left\{ W_{q_i} \sim N\left(W_{q_{i-1}} + \frac{(q_i - q_{i-1})}{(q_{n+1} - q_{i-1})}(W_{q_{n+1}} - W_{q_{i-1}}), \frac{(q_{n+1} - q_i)(q_i - q_{i-1})}{(q_{n+1} - q_{i-1})}\right)\right.$
return $(W_{q_1}, ..., W_{q_n})$

3.2 Diffusion Processes and Path–Space Rejection Sampling

In this section, we slightly divert our attention from Brownian motion to introduce the concept of a diffusion process and outline its path-space rejection sampling. At

first, we define a diffusion process as:

Definition 3.2.1 ***Diffusion processes:*** *A diffusion process* $\{\mathbf{X}_t : t \geq 0\}$, $\mathbf{X} : \mathbb{R} \mapsto \mathbb{R}^d$ *is a Markov process which satisfies the stochastic differential equation*

$$d\mathbf{X}_t = \beta(\mathbf{X}_t)dt + \Sigma(\mathbf{X}_t)d\mathbf{W}_t, \ \mathbf{X}_0 = \mathbf{x}_0, \tag{3.2.1}$$

where $\beta(\cdot)(d \times 1)$ *and* $\Sigma(\cdot)(d \times d)$ *denotes the drift and diffusion coefficient respectively. Furthermore,* $\mathbf{W}_t$ *denotes a* $d-$*dimensional Brownian motion.*

Some regularity conditions are assumed on the drift and the diffusion coefficients in order to ensure the existence of the solution of a stochastic differential equation (see [Kloeden and Platen, 1999] or [Øksendal, 2003] for more details).

3.2.1 Path-Space Rejection Sampling and Related Concepts

Here, we introduce a class of rejection mechanism to sample the trajectories of a given diffusion process. The *path-space rejection sampling* is a rejection mechanism on a diffusion path-space, over a fixed time interval ([Beskos and Roberts, 2005]; [Beskos et al., 2006, 2008]). Formally, let $\{\mathbf{X}_t : t \geq 0\}$ be a diffusion of interest defined on the probability space $(\mathbf{\Omega}, \mathcal{F}, \mathbb{Q})$. We are interested in simulating a sample path according to measure $\mathbb{Q}$ on the interval $[0, t]$. However, it might be difficult to simulate a sample path according to measure $\mathbb{Q}$, therefore, we choose an equivalent measure $\mathbb{W}$ (typically a Brownian motion) on the space $(\mathbf{\Omega}, \mathcal{F})$:

Definition 3.2.2 ***Equivalent measure:*** *A probability measure* $\mathbb{Q}$ *is equivalent to another measure* $\mathbb{W}$ *defined on* $(\mathbf{\Omega}, \mathcal{F})$ *if* $\mathbb{Q}(A) = 0$ *iff* $\mathbb{W}(A) = 0$*, for any measurable set* $A \in \mathcal{F}$*.*

The measure $\mathbb{W}$ is chosen such that it is easier to simulate sample paths according to $\mathbb{W}$ and the *Radon-Nikodým derivative* (see for instance [Øksendal, 2003]) is bounded. Formally, there exists $c > 0$ such that for any path $\mathbf{X} = \{\mathbf{X}_s : 0 \leq s \leq t\}$ over a bounded time interval, we have,

$$\frac{d\mathbb{Q}}{d\mathbb{W}}(\mathbf{X}) \leq c. \tag{3.2.2}$$

We then propose a sample path $\mathbf{X}$ according to the measure $\mathbb{W}$ and accept it with probability

$$P_{\mathbb{W}}(\mathbf{X}) := \frac{1}{c}\frac{d\mathbb{Q}}{d\mathbb{W}}(\mathbf{X}) \in [0, 1]. \tag{3.2.3}$$

Therefore, $\mathbf{X} \,|\, (accepted) \sim \mathbb{Q}$ and the average acceptance probability of the accepted sample path is given by $E_{\mathbb{Q}}(P_{\mathbb{W}}(\mathbf{X})) = \frac{1}{c}$. Therefore, a smaller value of c is preferred in condition (3.2.2) which ensures a tight bound and hence, a greater chance of the proposed sample path of being accepted.

However, the continuous-time simulation of a sample path of a diffusion process is impossible. This is impractical due to the computational cost of an infinite-dimensional object and its storage difficulty. As a result, we need to sample the trajectories of a diffusion process at a finite collection of time points. As introduced in Chapter (1), one approach to simulate the trajectory of (3.2.1) is the *time-discretization* method such as Euler-Maruyama [Kloeden and Platen, 1999]. In a time-discretization scheme we assume that the position of a trajectory at time $t + h$ can be approximated by a Gaussian density, namely

$$\mathbf{X}_{t+h} | (\mathbf{X}_t = \mathbf{x}_t) \sim N\left(\mathbf{x}_t + h\beta(\mathbf{x}_t), h\Sigma\Sigma^{\top}(\mathbf{x}_t)\right). \tag{3.2.4}$$

Therefore, we can break the time interval of the sample path into a fine mesh (each of small size h), then simulate the next position using the Euler-Maruyama approximation (3.2.4). The simulated sample path in this manner can be used as a proxy for an entire sample path drawn according to the target measure $\mathbb{Q}$. However, time-discretization methods such as Euler-Maruyama can lead to several issues to achieve a good approximation. Precisely, minimising the approximation error (making h very small) leads to an increase in the computational cost. Furthermore, discretization schemes create bias in the approximation which accumulates over time. Therefore, in this work we will resort to techniques which can be used to sample trajectories without any approximation error. A detailed account of such techniques can be found in [Beskos and Roberts, 2005; Beskos et al., 2006, 2008] and [Pollock, 2013].

Earlier in this section we reviewed the path-space rejection sampler to sample the trajectories of a diffusion process without discretization. However, it needs to be simulated at a finite collection of time points for it to be computationally feasible, and therefore, we define the notion of a *skeleton* which can be used to characterise the entire sample path of a diffusion.

Definition 3.2.3 ([Pollock, 2013]) *Skeleton:* *A skeleton* $\mathbf{S}$ *denotes a finite dimensional representation of the sample path of a diffusion which can be simulated without any approximation error. Typically, the path-space rejection sampling can*

be used to obtain a sample path $\mathbf{X} = \{\mathbf{X}_s : 0 \leq s \leq t\}$ *at a finite collection of time points, where a proposal sample path is drawn according to an equivalent measure* $\mathbb{W}$ *and accepted for the target measure* $\mathbb{Q}$ *with a probability proportional to* $\frac{d\mathbb{Q}}{d\mathbb{W}}(\mathbf{X})$.

We require the bounding condition (3.2.2) on the Radon-Nikodým derivative to hold in order to obtain a skeleton of the sample path. Furthermore, this makes sure that the sample path can be obtained with a finite computational cost. However, we should note that in a more general setting the bounding constant c in (3.2.2) can be very large or even $+\infty$. To circumvent this problem, one can restrict the sample path $\mathbf{X} \sim \mathbb{W}$ to a compact interval such that the Radon-Nikodým derivative is almost surely bounded. This leads to the notion of partitioning of the path-space of a proposal measure $\mathbb{W}$ into distinct *layers* such that the sample path can be simulated based on which layer it belongs to.

Definition 3.2.4 ([Pollock, 2013]) ***Layer information:*** *A layer information* $R(\mathbf{X})$ *(where* $\mathbf{X}$ *is simulated according to the proposal measure* $\mathbb{W}$*) for a diffusion process is a function of its sample trajectory that determines the interval to which its sample trajectories* $\mathbf{X}$ *are constrained.*

Partitioning the path-space into sets of layers requires us to be able to simulate the skeleton according to the proposal measure $\mathbb{W}$, conditional upon the layer information R. A simulation mechanism in the case of a Brownian motion has been outlined in Section (3.4).

3.3 Simulation of Brownian Motion using Localisation

An alternative mechanism of simulating the sample path of a Brownian motion often called the *Localisation* approach appeared in a number of literatures (for instance [Chen and Huang, 2013], [Pollock, 2013] and [Pollock et al., 2017]). This method allows us to simulate the trajectories of a Brownian motion exactly under various restrictions, without suffering from any discretization error, which is a key ingredient in the construction of the methodology developed in this work. Therefore, we briefly outline its construction in the following paragraph.

Recall that Algorithm (3.1.1) allows us to simulate the path of a Brownian motion at user-specified time points $\{t_i\}_{i\geq 1}$ in the interval $[0, t]$, by successively simulating W_{t_i} from a normal distribution with mean $W_{t_{i-1}}$ and variance $(t_i - t_{i-1})$. Alternatively, under the localisation approach we break the sample path of a Brownian motion into a number of bounded segments. Here, each segment can be successively

simulated by simulating a sequence of first passage times τ, for some user specified boundary $\theta > 0$ symmetric around its starting point $W_0 = x$, therefore,

$$\tau := \inf\{s > 0 \, : \, W_s \notin [x - \theta, x + \theta]\}. \tag{3.3.1}$$

This idea has been illustrated in Figure (3.1) where a sequence of first passage times

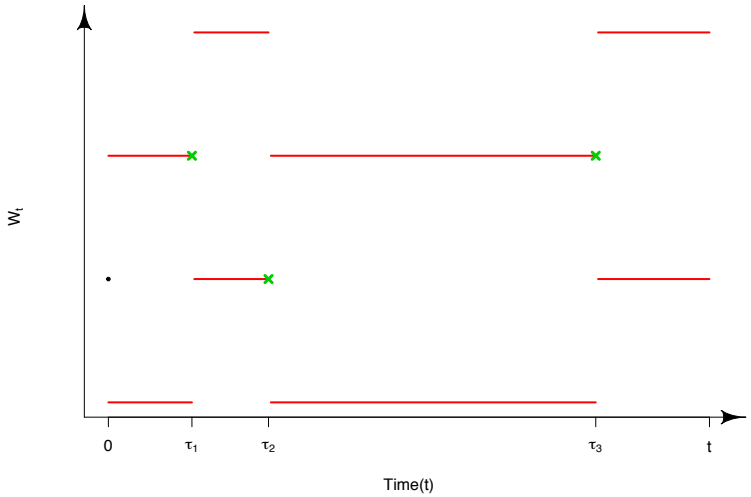


Figure 3.1: *This figure depicts the one of the stages of localisation process. For a user specified boundary size θ, a sequence of exit times are simulated. Note that the trajectory of Brownian motion is desired until time t.*

$$\tau_i = \tau_{i-1} + \inf\{s \geq 0 \, : \, W_s \notin [W_{\tau_{i-1}} - \theta, W_{\tau_{i-1}} + \theta]\}, \tau_0 = 0 \tag{3.3.2}$$

are simulated until time t – the time until which the trajectory of Brownian motion is desired. Here, we denote each collection of the first passage information by the *layer information* $R_W^{(i)} = (\tau_i, W_{\tau_{i-1}})$. This is followed by the simulation of positions of a Brownian motion W_{t_i} at the user-specified time points $t_i \in [\tau_{j-1}, \tau_j]$ for $i, j \geq 1$, conditional on the layer information $R_W^{(i)}$ such that the sample path is constrained within the bounds $[W_{\tau_{i-1}} - \theta, W_{\tau_{i-1}} + \theta]$. Figure (3.2) illustrates the idea of construction of a sample path of a Brownian motion at a sequence of user-specified time points.

In a more general setting where we are interested in simulating the trajectories of a $d-$dimensional Brownian motion $\{\mathbf{W}_t = (W_t^{(1)}, ..., W_t^{(d)})\}_{t \geq 0}$ initialised at $\mathbf{W}_0$; we can achieve this by first specifying a hypercube of side lengths $2\theta^{(1)}, ..., 2\theta^{(d)}$

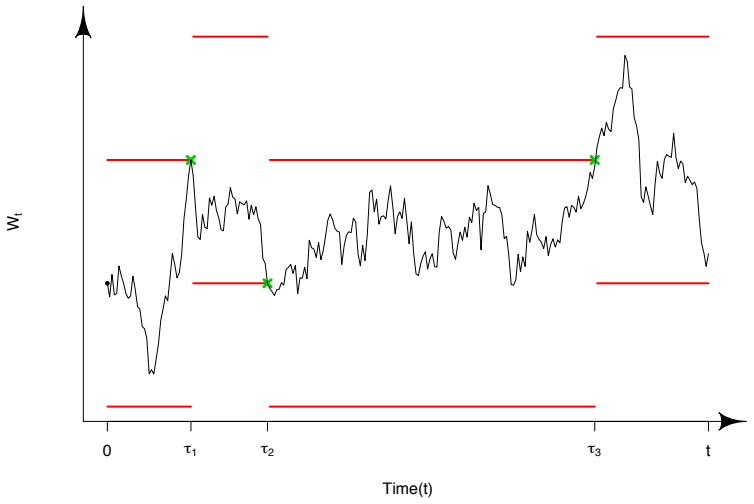


Figure 3.2: *This figure depicts the second stage of the localisation process. The simulation of a sample path at a sequence of user-specified times are achieved by simulating according to the law of a Brownian motion conditional on the layer information, together with the sample path being constrained within the bounds. Note that the trajectory of Brownian motion is desired until time t.*

around $\mathbf{W}_0$, which is defined in the following sense:

Definition 3.3.1 *A $d-$dimensional hypercube $\mathcal{H}$ around the point $\mathbf{x}_0 = (x_0^{(1)}, ..., x_0^{(d)}) \in \mathbb{R}^d$ with its side lengths $2\theta^{(1)}, ..., 2\theta^{(d)}$ is*

$$\mathcal{H} := [x_0^{(1)} - \theta^{(1)}, x_0^{(1)} + \theta^{(1)}] \times ..., \times [x_0^{(d)} - \theta^{(d)}, x_0^{(d)} + \theta^{(d)}]. \tag{3.3.3}$$

As outlined in the univariate setting, we simulate the time $\tau_1 := \inf\{t \geq 0 : \mathbf{W}_t \notin \mathcal{H}\}$ and the position of the first passage $\mathbf{W}_{\tau_1}$ of a Brownian motion from this hypercube. Next, a new hypercube is specified around the position of the first passage $\mathbf{W}_{\tau_1}$ of the preceding hypercube and we again simulate the time τ_2 and the position of the first passage $\mathbf{W}_{\tau_2}$ of a Brownian motion from this hypercube. This process is repeated until a fixed time $t > 0$. Similarly, the intermediate positions of a Brownian motion can be filled using the conditional law of the process, conditional on its layer information for a given hypercube. Note that in order to construct the sample path of a Brownian motion at a user-specified time, we need to be able to construct the following:

- simulate the first passage time τ and the position of a Brownian motion $\mathbf{W}_\tau$ from a given hypercube $\mathcal{H}$.

- Simulate the position of a Brownian motion at time q conditional on the fact that it achieves an extrema at $(\tau, \mathbf{W}_\tau)$, conditional on the fact that it is constrained within a given hypercube $\mathcal{H}$.

We address the simulation of the points mentioned above in further sections that follow within this chapter.

3.4 Simulation of the First Passage Time of a Standard Brownian Motion

As outlined in the previous section, the simulation of the first passage time of a standard Brownian motion is of profound interest pertaining to the localisation method of a Brownian motion. This is a key ingredient in the simulation of trajectories of a Brownian motion and therefore, we describe a methodology to simulate the layer information in this section.

Burq and Jones [2008] first presented the methodology to simulate the first passage time of a univariate Brownian motion for a given symmetric boundary (i.e. a symmetric hypercube around the point under consideration). This approach can be further extended to construct the first passage time of a $d-$dimensional standard Brownian motion for a symmetric boundary. For notational convenience we denote a $d-$dimensional standard Brownian motion by $\{\mathbf{W}_t = (W_t^{(1)}, ..., W_t^{(d)})\}_{t\geq 0}$ initialised at $\mathbf{W}_0$ and a symmetric hypercube around $\mathbf{W}_0$ denoted by $\mathcal{H} = [W_0^{(1)} - \theta^{(1)}, W_0^{(1)} + \theta^{(1)}] \times ... \times [W_0^{(d)} - \theta^{(d)}, W_0^{(d)} + \theta^{(d)}]$. At first, we restrict ourselves to the (i^{th}) dimension of a Brownian motion $\{W_t^{(i)}\}_{t\geq 0}$ exiting the boundary $[W_0^{(i)} - \theta^{(i)}, W_0^{(i)} + \theta^{(i)}]$ and postpone the discussion of the first passage time in a $d-$ dimensional case until Section (3.4.2). The first passage time for the (i^{th}) dimension is given by

$$\tau^{(i)} = \inf\{t \geq 0 : |W_t^{(i)} - W_0^{(i)}| \geq \theta^{(i)}\}. \tag{3.4.1}$$

Using the scaling property (Property (3.1.1)) of a standard Brownian motion, it can be inferred that the law of $\{W_t^{(i)}\}_{t\geq 0}$ is identical to the law of the process $\{\theta^{(i)} W^{(i)}_{t/\theta^{(i)2}}\}_{t\geq 0}$ [Karatzas and Shreve, 1991]. Therefore, the first passage time $\tau^{(i)}$ of a univariate Brownian motion exiting the boundary $[W_0^{(i)} - \theta^{(i)}, W_0^{(i)} + \theta^{(i)}]$ satisfies

$$\tau^{(i)} = \left(\theta^{(i)}\right)^2 \bar{\tau}. \tag{3.4.2}$$

Here $\bar{\tau} := \inf\{t \geq 0 : |W_t - W_0| \geq 1\}$ is the first passage time of a univariate Brownian motion starting at W_0 and exiting the boundary $[W_0 - 1, W_0 + 1]$. Therefore, the sampling of $\tau^{(i)}$ can be achieved by the simulation of $\bar{\tau}$, if we knew how to sample $\bar{\tau}$. We denote the density of $\bar{\tau}$ by $f_{\bar{\tau}}$; the analytic form of this probability density function is given as [Burq and Jones, 2008]

$$f_{\bar{\tau}}(t) = \sum_{k=-\infty}^{\infty} (-1)^k \frac{(2k+1)}{\sqrt{2\pi t^3}} \exp\left(-\frac{(2k+1)^2}{2t}\right) \quad \text{for} \quad t \geq 0. \tag{3.4.3}$$

Further, [Burq and Jones, 2008] showed that the density $f_{\bar{\tau}}$ can be bounded above by a gamma density up to a constant. Formally, there exists constants a, b and λ such that

$$f_{\bar{\tau}}(t) \leq a g(t \,|\, b, \lambda) \quad \text{for all} \quad t \geq 0, \tag{3.4.4}$$

where $g(t \,|\, b, \lambda) = \frac{\lambda^b}{\Gamma(b)} t^{b-1} \exp(-\lambda t)$ is the density of a gamma distribution with parameters b and λ which is evaluated at a point $t > 0$. This information can be used to construct a rejection sampling algorithm to simulate the first passage time; this is discussed in Section (3.4.1).

3.4.1 A Rejection Sampling Scheme to Simulate Standard First Passage Time

[Burq and Jones, 2008] numerically showed that the optimal values of a, b and λ are given by $a = 1.243707, b = 1.088870$ and $\lambda = 1.233701$. The inequality (3.4.4) allows us to use a rejection sampling scheme (ref to Section (2.1) for Algorithm (2.1.1)) to simulate values according to the density $f_{\bar{\tau}}$, if this density can be evaluated exactly. In such a setting, we could generate proposal samples $V \sim g(\cdot \,|\, b, \lambda)$ according to a gamma density and accept it for the target density $f_{\bar{\tau}}$ with probability $f_{\bar{\tau}}(V)/(a \times g(V \,|\, b, \lambda))$. The decision on the acceptance/rejection of a proposed point V can be made by generating a uniform random number $U \sim U[0, 1]$ and accepting if $U(a \times g(V \,|\, b, \lambda)) < f_{\bar{\tau}}(V)$. The exact evaluation of the density $f_{\bar{\tau}}$ is impossible due to the infinite sum and hence, the comparison $U(a \times g(V \,|\, b, \lambda)) < f_{\bar{\tau}}(V)$ is impractical. However, special characteristic of the sum (3.4.3) can be used to perform the comparison. [Burq and Jones, 2008] have proved that the partial sum defined by

$$f_{\bar{\tau}}^{(J)}(t) := \sum_{k=-J}^{J} (-1)^k \frac{(2k+1)}{\sqrt{2\pi t^3}} \exp\left(-\frac{(2k+1)^2}{2t}\right) \tag{3.4.5}$$

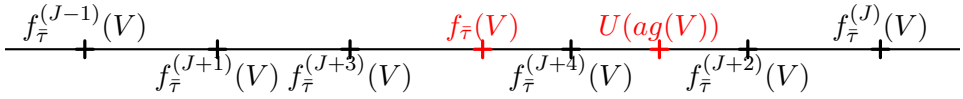


Figure 3.3: *This figure illustrates the oscillating nature of the sequence* $f_{\bar{\tau}}^{(J)}(V)$. *It should be noted that the sequence* $f_{\bar{\tau}}^{(J-1)}(V), f_{\bar{\tau}}^{(J+1)}(V), f_{\bar{\tau}}^{(J+3)}(V), \ldots$ *converges to* $f_{\bar{\tau}}(V)$ *from below while the sequence* $f_{\bar{\tau}}^{(J)}(V), f_{\bar{\tau}}^{(J+2)}(V), f_{\bar{\tau}}^{(J+4)}(V), \ldots$ *converges from above. Thus* $(U(ag(V)) - f_{\bar{\tau}}^{(J)}(V))$ *and* $(U(ag(V)) - f_{\bar{\tau}}^{(J+1)}(V))$ *will have differing signs until the comparison stops.*

has the *oscillating property*, which can be used to perform the inequality comparison without having to calculate the infinite sum explicitly. [Burq and Jones, 2008, Lemma 4] shows that the sequence of partial sums $f_{\bar{\tau}}^{(J)}(t)$ are oscillating for $J \geq \max(t\log(3)/4, 3)$. Under the accept–reject mechanism, our main aim is to compare $f_{\bar{\tau}}(V)$ with $U(ag(V \mid b, \lambda))$ upon proposing $V \sim g(\cdot \mid b, \lambda)$ and simulating $U \sim U[0,1]$. We can directly employ the alternating series sampling Algorithm (2.4.2) in order to perform acceptance/rejection on the proposed point V.

Therefore, for a given level $\theta^{(i)}$, the first passage time of a univariate Brownian motion is given by $\tau^{(i)} = (\theta^{(i)})^2 V$, while the position $W_{\tau^{(i)}}^{(i)}$ at the first passage time satisfies

$$\mathbb{P}\left(W_{\tau^{(i)}}^{(i)} = W_0^{(i)} + \theta^{(i)}\right) = \mathbb{P}\left(W_{\tau^{(i)}}^{(i)} = W_0^{(i)} - \theta^{(i)}\right) = \frac{1}{2}. \tag{3.4.6}$$

Thus, we set the exit position $W_0^{(i)} + \theta^{(i)}$ and $W_0^{(i)} - \theta^{(i)}$ with a probability of 0.5. The algorithm to simulate the first passage time and its position has been synthesised in Algorithm (3.4.1). Figure (3.4) plots the kernel density estimator of the first passage time of a univariate standard Brownian motion starting at 0 and

exiting the boundary $[-1, 1]$.

Algorithm 3.4.1: FIRST PASSAGE TIME SIMULATOR$(W_0^{(i)}, \theta^{(i)})$

output $(\tau^{(i)}, W_{\tau^{(i)}}^{(i)})$
global $a \leftarrow 1.243707; b \leftarrow 1.088870; \lambda \leftarrow 1.233701$
$N(t) \leftarrow \max\{t \log(3)/4, 3\}$;
$V \sim$ ALTERNATING SERIES SAMPLER$(f, a, Gamma(\cdot, b, \lambda), N(\cdot))$
$U \sim U[0, 1]$
$\tau^{(i)} \leftarrow (\theta^{(i)})^2 V$
if $U \leq 0.5$
  then $W_{\tau^{(i)}}^{(i)} \leftarrow W_0^{(i)} + \theta^{(i)}$
  else $W_{\tau^{(i)}}^{(i)} \leftarrow W_0^{(i)} - \theta^{(i)}$
return $(\tau^{(i)}, W_{\tau^{(i)}}^{(i)})$

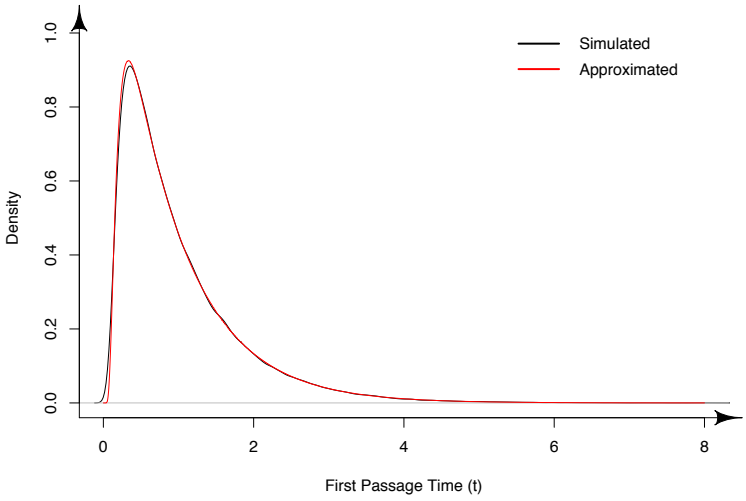


Figure 3.4: *This figure plots the kernel density estimator of the first passage time (based on 2×10^5 samples) of a standard Brownian motion starting at 0 and exiting the boundary $[-1, 1]$. The red curve is the numerically approximated density using $f_{\bar{\tau}}^{(J)}$ for $J = 10^5$.*

3.4.2 Simulating the First Passage Time and Position of a $d-$dimensional Brownian Motion

In Algorithm (3.4.1) we presented the algorithm that simulates the first passage time for a univariate Brownian motion. This algorithm can be extended to sample the

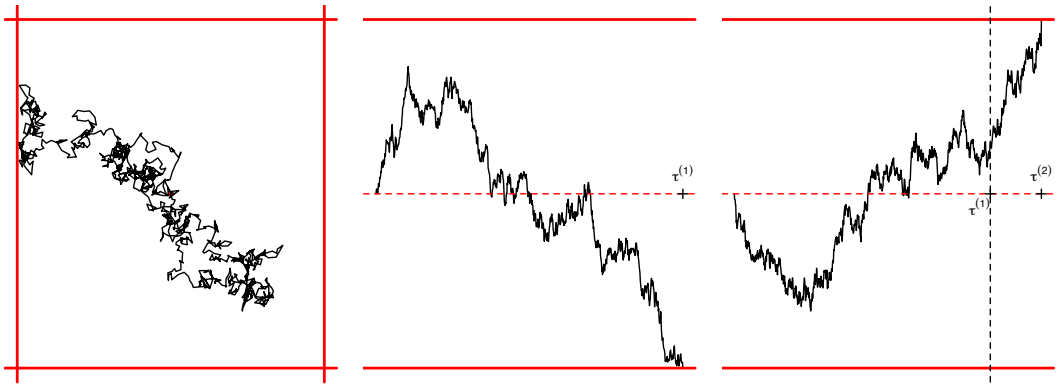


Figure 3.5: *An illustration of a 2-dimensional Brownian motion exiting a symmetric hypercube around* $(0,0)$ *(left). It should be noted that a Brownian motion exits along the* x*-dimension first* $(\tau^{(1)} < \tau^{(2)})$ *through the lower boundary (middle). The exit position of the Brownian motion along the* y*-dimension is given by its position at* $\tau^{(1)}$ *i.e.* $W^{(2)}_{\tau^{(1)}}$ *(right). In this context, the position* $W^{(2)}_{\tau^{(1)}}$ *is obtained using the law of a Brownian bridge attaining a maximum at* $\tau^{(2)}$ *and conditioned to remain bounded.*

first passage time of a $d-$dimensional Brownian motion $\{\mathbf{W}_t = (W_t^{(1)}, ..., W_t^{(d)})\}_{t\geq 0}$ initialised at $\mathbf{W}_0$, exiting a symmetric hypercube around $\mathbf{W}_0$ denoted by $\mathcal{H} = [W_0^{(1)} - \theta^{(1)}, W_0^{(1)} + \theta^{(1)}] \times ... \times [W_0^{(d)} - \theta^{(d)}, W_0^{(d)} + \theta^{(d)}]$. In order to simulate the first passage time for a $d-$dimensional Brownian motion, we first sample the exit times $\tau^{(1)}, ..., \tau^{(d)}$ for each dimension of $\mathbf{W}_t$. The first passage time τ from the hypercube $\mathcal{H}$ is given by the minimum value of $\tau^{(1)}, ..., \tau^{(d)}$. Formally,

$$\tau = \min\{\tau^{(1)}, ..., \tau^{(d)}\}. \tag{3.4.7}$$

Next, our aim is to find the position of the Brownian motion at the first passage time τ. We proceed as follows. Let m denote the dimension along which the minimum of $\tau^{(i)}$'s is obtained, i.e.

$$\tau = \tau^{(m)} \quad \text{for some} \quad m \in \{1, ..., d\}. \tag{3.4.8}$$

Along the m^{th} dimension, a Brownian motion touches its boundary, which implies that either

$$W_\tau^{(m)} = W_0^{(m)} - \theta^{(m)} \quad \text{or} \quad W_\tau^{(m)} = W_0^{(m)} + \theta^{(m)}. \tag{3.4.9}$$

For other dimensions $j \in \{1, ..., d\} \setminus \{m\}$, it follows that

$$W_0^{(j)} - \theta^{(j)} < W_\tau^{(j)} < W_0^{(j)} + \theta^{(j)}. \tag{3.4.10}$$

We further note that for every $j \in \{1, ..., d\} \setminus \{m\}$, the exit time satisfies $\tau^{(j)} > \tau$. Therefore, the position of a Brownian motion at time τ along the $j-$th dimension $W_\tau^{(j)}$ is obtained using the law of a Brownian bridge process starting at $(0, W_0^{(j)})$ and terminating at $(\tau^{(j)}, W_{\tau^{(j)}}^{(j)})$, conditioned to remain inside the bounds $[W_0^{(j)} - \theta^{(j)}, W_0^{(j)} + \theta^{(j)}]$. We refer to Section (3.6) for a simulation algorithm from such a process. An illustration for a standard two-dimensional Brownian motion is given in the Figure (3.5).

Algorithm 3.4.2: MULTIVARIATE FIRST PASSAGE TIME SIMULATOR$(\mathbf{W}_0, \theta^{(1)}, ..., \theta^{(d)})$

---

output $(\tau, \mathbf{W}_\tau)$
$\tau \leftarrow \infty$
for $i \leftarrow 1$ **to** d
  do $\begin{cases} (\tau^{(i)}, W_{\tau^{(i)}}^{(i)}) \leftarrow \text{FIRST PASSAGE TIME SIMULATOR}(W_0^{(i)}, \theta^{(i)}) \\ \tau \leftarrow \min\{\tau, \tau^{(i)}\} \end{cases}$
$m \leftarrow \{i : \tau = \tau^{(i)}\}$
for each $j \in \{1, ..., d\} \setminus \{m\}$
  do $W_\tau^{(j)} \leftarrow$ BESSEL BRIDGE SIMULATOR$(\tau, 0, W_0^{(j)}, \tau^{(j)}, W_{\tau^{(j)}}^{(j)}, \theta^{(j)})$(see Section (3.5))
return $(\tau, \mathbf{W}_\tau)$

3.5 Simulation of a Brownian Motion Given it Attains a Extremum which is Constrained within an Interval

In Section (3.4) we encountered a situation where it was required to simulate the position of the j^{th} dimension of a Brownian motion at an instance τ such that $\tau < \tau^{(j)}$. This requires us to be able to simulate the position of a Brownian motion given its first passage information and it is constrained to stay within a specified bound. Therefore, in this section we outline a method to simulate the position of a Brownian motion, given its initial position and first passage information, which is constrained within a given hypercube.

In our case, each of the dimensions of a $d-$dimensional Brownian motion are independent and hence, each dimension of a Brownian motion can be treated separately.

Therefore, we restrict ourselves to a situation of simulating the intermediate position W_q for $q \in (s, \tau)$, given its positions W_s, W_τ at s and τ respectively, which is constrained to remain in the interval $[W_s - \theta, W_s + \theta]$. We observe that the path of a univariate Brownian motion $W_{q\{s \leq q \leq \tau\}}$ initialised at (s, W_s) which attains its extremal value at (τ, W_τ) follows the law of a 3−dimensional Bessel bridge [Pollock, 2013; Pollock et al., 2017]. Moreover, the path of a 3−dimensional Bessel bridge can be decomposed into three independent Brownian bridges. Recall that a d–dimensional Bessel process is defined as the Euclidean norm of a d–dimensional standard Brownian motion [Øksendal, 2003]. We then require that the simulated position W_q is constrained to the interval $[W_s - \theta, W_s + \theta]$ for all $q \in [s, \tau]$. This can be achieved using a rejection sampling mechanism in which a Bessel bridge sample path W_q is simulated at time q and the position W_q is accepted if $\{W_t\}_{\{s \leq t \leq q\}} \in [W_s - \theta, W_s + \theta]$ and $\{W_t\}_{\{q \leq t \leq \tau\}} \in [W_s - \theta, W_s + \theta]$. Let $\{b^{(i)}\}_{i=1,2,3}$ denote three independent Brownian bridges of unit length between $(0, 0)$ and $(1, 0)$. The realised position of the Bessel bridge sample path at $q \in [s, \tau]$ is given by

$$W_q = \begin{cases} W_s - \left\{(\tau - s)\left[\left(b_q^{(1)} + \frac{\theta(\tau - q)}{(\tau - s)^{1.5}}\right)^2 + \left(b_q^{(2)}\right)^2 + \left(b_q^{(3)}\right)^2\right]\right\}^{\frac{1}{2}} & \text{if} \quad W_\tau = W_s - \theta \\ W_s + \left\{(\tau - s)\left[\left(b_q^{(1)} + \frac{\theta(\tau - q)}{(\tau - s)^{1.5}}\right)^2 + \left(b_q^{(2)}\right)^2 + \left(b_q^{(3)}\right)^2\right]\right\}^{\frac{1}{2}} & \text{if} \quad W_\tau = W_s + \theta \end{cases} \tag{3.5.1}$$

Next, a decision on the acceptance/rejection of a proposed point W_q is made based on whether or not W_q lies in the interval $[W_s - \theta, W_s + \theta]$. This requires us to evaluate the probability p such that the sample trajectories $\{W_t\}_{\{s \leq t \leq q\}} \in [W_s - \theta, W_s + \theta]$ and $\{W_t\}_{\{q \leq t \leq \tau\}} \in [W_s - \theta, W_s + \theta]$. [Pollock et al., 2017, Theorem 3] gives an analytic form of the probability p which is a product of two infinite series. We present the result in the Theorem (3.5.1).

Theorem 3.5.1 ([Pollock et al., 2017]) *Let* $\mathbb{W}_{s,\tau}^{W_s, W_\tau} \mid (W_q, W_\tau)$ *for some* $q \in [s, \tau]$ *denote the law of a three-dimensional Bessel bridge process attaining its extremal value at* (τ, W_τ)*, constrained in the interval* $[W_s - \theta, W_s + \theta]$*. If* $\{W_t\}_{\{s \leq t \leq \tau\}} \sim \mathbb{W}_{s,\tau}^{W_s, W_\tau} \mid (W_q, W_\tau)$ *denotes the sample trajectory in the interval* $[s, \tau]$ *and* $m := \mathbb{I}(W_\tau > W_s) - \mathbb{I}(W_\tau < W_s)$*, then the probability that* $\{W_t\}_{\{s \leq t \leq \tau\}}$ *remains restricted*

to the interval $[W_s - \theta, W_s + \theta]$ *is given by*

$$\mathbb{P}(\{W_t\}_{\{s\leq t\leq \tau\}} \in [W_s-\theta, W_s+\theta] \,|\, (W_s, W_q, W_\tau)) = \left(\frac{1 - \sum_{j=1}^{\infty} \left(\zeta_{q-s}(j; W_s - W_q, \theta) - \varphi_{q-s}(j; W_s - W_q, \theta)\right)}{1 - \exp\left(-2\theta(m(W_s - W_q) + \theta)/(q-s)\right)}\right) \times \left(1 + \sum_{j=1}^{\infty} \left(\phi_{\tau-q}(j; W_q - W_\tau, \theta, m) + \chi_{\tau-q}(j; W_q - W_\tau, \theta, m)\right)\right), \tag{3.5.2}$$

where

$$\zeta_\Delta(j; \delta, \theta) := 2\exp\left(-\frac{2\theta^2(2j-1)^2}{\Delta}\right)\cosh\left(\frac{2(2j-1)\theta\delta}{\Delta}\right), \tag{3.5.3}$$

$$\varphi_\Delta(j; \delta, \theta) := 2\exp\left(-\frac{8\theta^2 j^2}{\Delta}\right)\cosh\left(\frac{4\theta\delta j}{\Delta}\right), \tag{3.5.4}$$

$$\phi_\Delta(j; \delta, \theta, m) := \chi_\Delta(j; \delta, \theta, m) := \frac{(4\theta j + m\delta)}{m\delta}\exp\left(-\frac{4\theta j(2\theta j + m\delta)}{\Delta}\right). \tag{3.5.5}$$

In an accept–reject setting, the proposed point W_q can be accepted with probability p given by Theorem (3.5.1). To achieve this we can simulate $U \sim U[0,1]$ and make a decision based on $U \leq p$ and $U > p$. However, due to the infinite sum in the expression of p the comparison is not possible. To circumvent this, a series sampler (2.4) can be used to perform the inequality comparison without having to calculate p exactly [Pollock et al., 2017; Pollock, 2013]. To achieve this, Pollock et al. [2017] argues that a monotonically convergent upper and lower sequence of partial sums can be constructed such that they converge from the above and below respectively. Therefore, monotonically convergent sequences $p_n^{\uparrow}, p_n^{\downarrow}$ can be constructed which satisfies

$$\lim_{n\to\infty} p_n^{\uparrow} \to p \quad \text{and} \quad \lim_{n\to\infty} p_n^{\downarrow} \to p, \tag{3.5.6}$$

with the property that for all $U \in [0,1]$ and $\epsilon > 0$, $\exists\, N(t)$ such that $\forall\, n \geq N(t)$,

$$p_n^{\uparrow} - p_n^{\downarrow} < \epsilon. \tag{3.5.7}$$

Expression (3.5.7) guarantees that a decision on the acceptance/rejection can be made up to an arbitrarily high precision with a finite computational effort. [Pollock

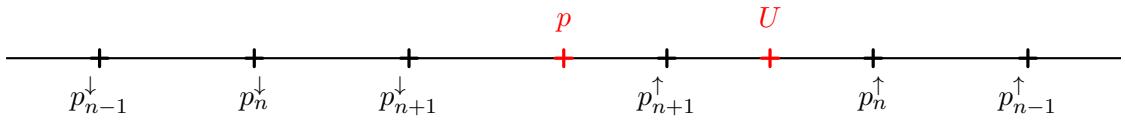


Figure 3.6: *The upper and the lower converging sequences of probabilities which converge to the required probability p.*

et al., 2017, Corollary 1] can be used to construct the upper and lower bounding sequences of probabilities which is presented in the Corollary (3.5.2).

Corollary 3.5.2 ([Pollock et al., 2017]) *Let* $p := \mathbb{P}(\{W_t\}_{\{s\leq t\leq \tau\}} \in [W_s-\theta, W_s+\theta])$ *and* $N(q) \geq \lceil \sqrt{(\tau-q)+4\theta^2}/4\theta \rceil$. *Its monotonically converging upper* $p^{\uparrow}_n$ *and lower* $p^{\downarrow}_n$ *sequence of probabilities can be constructed as*

$$p^{\downarrow}_n := \left(\frac{1 - \sum_{j=1}^{n} \zeta_{q-s}(j; W_s - W_q, \theta) + \sum_{j=1}^{n-1} \varphi_{q-s}(j; W_s - W_q, \theta)}{1 - \exp\left(-2\theta(m(W_s - W_q) + \theta)/(q-s)\right)} \right) \times \left(1 + \sum_{j=1}^{N(q)+n} \phi_{\tau-q}(j; W_q - W_\tau, \theta, m) + \sum_{j=1}^{N(q)+n-1} \chi_{\tau-q}(j; W_q - W_\tau, \theta, m) \right), \tag{3.5.8}$$

$$p^{\uparrow}_n := \left(\frac{1 - \sum_{j=1}^{n} \zeta_{q-s}(j; W_s - W_q, \theta) + \sum_{j=1}^{n} \varphi_{q-s}(j; W_s - W_q, \theta)}{1 - \exp\left(-2\theta(m(W_s - W_q) + \theta)/(q-s)\right)} \right) \times \left(1 + \sum_{j=1}^{N(q)+n} \phi_{\tau-q}(j; W_q - W_\tau, \theta, m) + \sum_{j=1}^{N(q)+n} \chi_{\tau-q}(j; W_q - W_\tau, \theta, m) \right). \tag{3.5.9}$$

Noting that the sequences $p_n^{\downarrow}$ & $p_n^{\uparrow}$ converge to the required probability p, we have

$$p_n^{\downarrow} < p_n^{\uparrow} < U \implies p < U \quad \text{and} \tag{3.5.10}$$

$$U < p_n^{\downarrow} < p_n^{\uparrow} \implies p > U \quad \text{for some } n. \tag{3.5.11}$$

Based on comparisons (3.5.10) and (3.5.11), it can be implied that a decision on acceptance/rejection will not be made as long as $p_n^{\downarrow} < U < p_n^{\uparrow}$. Therefore, we continue incrementing the value of n as long as U falls between $p_n^{\downarrow}$ and $p_n^{\uparrow}$. Once $U \notin [p_n^{\downarrow}, p_n^{\uparrow}]$ for some n, a decision is made based on the inequalities (3.5.10) or (3.5.11).

Algorithm 3.5.1: BESSEL BRIDGE SIMULATOR$(q, s, W_s, \tau, W_\tau, \theta)$

output (W_q)
accepted $\leftarrow$ **false**
while accepted $\neq$ **true**
do
- $b_q^{(1)}, b_q^{(2)}, b_q^{(3} \sim N\left(0, \frac{(q-s)(\tau-q)}{(\tau-s)}\right)$
- $W_q \leftarrow W_s + (-1)^{\mathbb{I}(W_\tau < W_s)} \left\{(\tau - s)\left[\left(b_q^{(1)} + \frac{\theta(\tau-q)}{(\tau-s)^{1.5}}\right)^2 + \left(b_q^{(2)}\right)^2 + \left(b_q^{(3)}\right)^2\right]\right\}^{\frac{1}{2}}$
- $U \sim U[0,1]$ and $n = 1$
- **while** $p_n^{\downarrow} < U < p_n^{\uparrow}$
  - **do** $n = n + 1$
- **if** $U \leq p_n^{\downarrow}$
  - **then** accepted $\leftarrow$ **true**

return (W_q)

Once the position of Brownian motion W_q for $s < q < \tau$ has been realised, a user might be interested in simulating the position of Brownian motion W_ξ at a time point $s < \xi < q < \tau$. This situation is also of profound interest pertaining to the algorithm being developed in this work. Therefore, we consider an efficient method to address this problem in the following Section (3.6).

3.6 Simulation of a Brownian Bridge Position which is Constrained within an Interval

In this section we are interested in simulating the positions of a Brownian motion which is constrained within a given hypercube while it does not touch any side of

the hypercube. In Chapter (6) this situation is of profound interest to simulate the *regenerating position* of a Brownian motion once it has been *killed*. In our case, since each dimension of a Brownian motion is independent they can be treated separately. Therefore, we restrict our attention to simulating the position of a Brownian motion W_q for $q \in (q_1, q_2)$, given its positions W_{q_1} and W_{q_2}, which is strictly constrained within the interval $[W_s - \theta, W_s + \theta]$ (for $W_s - \theta < W_{q_1}, W_{q_2} < W_s + \theta$). The required position W_q can be proposed using the law of a Brownian bridge between (q_1, W_{q_1}) and (q_2, W_{q_2}) and accepted when it satisfies $W_q \in [W_s - \theta, W_s + \theta]$. Similar to arguments presented in Section (3.5), the proposed point W_q is accepted if and only if any sample path in the interval $[q_1, q]$ and $[q, q_2]$ satisfies $\{W_t\}_{\{q_1 \le t \le q\}} \in [W_s - \theta, W_s + \theta]$ and $\{W_t\}_{\{q \le t \le q_2\}} \in [W_s - \theta, W_s + \theta]$ respectively. Hence, we use the result pertaining to the probability that a Brownian bridge sample path is constrained within the interval $[W_s - \theta, W_s + \theta]$ [Pötzelberger and Wang, 2001]. We present this result in Theorem (3.6.1).

Theorem 3.6.1 ([Pötzelberger and Wang, 2001]) *Let* $\mathbb{W}_{q_1,q_2}^{W_{q_1},W_{q_2}} \,|\, (W_{q_1}, W_{q_2})$ *for some* $q \in [q_1, q_2]$ *denote the law of a Brownian bridge process constrained in the interval* $[W_s - \theta, W_s + \theta]$. *If* $\{W_t\}_{\{q_1 \le t \le q_2\}} \sim \mathbb{W}_{q_1,q_2}^{W_{q_1},W_{q_2}} \,|\, (W_{q_1}, W_{q_2})$ *denotes the sample trajectory in the interval* $[q_1, q_2]$ *then the probability that* $\{W_t\}_{\{q_1 \le t \le q_2\}}$ *remains restricted to the interval* $[W_s - \theta, W_s + \theta]$ *is given by*

$$\begin{aligned}\mathbb{P}(\{W_t\}_{\{q_1 \le t \le q_2\}} \in & [W_s - \theta, W_s + \theta] \,|\, (W_{q_1}, W_{q_2})) = \\ & \left(1 - \sum_{j=1}^{\infty} \left(\zeta_{q-q_1}(j; W_{q_1} - W_q, \theta) - \varphi_{q-q_1}(j; W_{q_1} - W_q, \theta)\right)\right) \times \\ & \left(1 - \sum_{j=1}^{\infty} \left(\zeta_{q_2-q}(j; W_q - W_{q_2}, \theta) - \varphi_{q_2-q}(j; W_q - W_{q_2}, \theta)\right)\right),\end{aligned} \tag{3.6.1}$$

where

$$\zeta_{\Delta}(j; \delta, \theta) := 2 \exp\left(-\frac{2\theta^2(2j-1)^2}{\Delta}\right) \cosh\left(\frac{2(2j-1)\theta\delta}{\Delta}\right), \tag{3.6.2}$$

$$\varphi_{\Delta}(j; \delta, \theta) := 2 \exp\left(-\frac{8\theta^2 j^2}{\Delta}\right) \cosh\left(\frac{4\theta\delta j}{\Delta}\right). \tag{3.6.3}$$

The proposed point W_q can be accepted with probability p given by Theorem (3.6.1). A decision on the proposed point W_q is made by simulating $U \sim U[0, 1]$ and accepting it if $U < p$. However, an exact evaluation of p is not possible due to the

infinite sum in the expression of p. Therefore, a series sampling algorithm (2.1) can use used to perform the comparison. We use [Pollock, 2013, Corollary 6.1.1] to construct a monotonically convergent upper and lower sequence of partial sums $p_n^{\uparrow}$ and $p_n^{\downarrow}$ respectively. We present the construction of $p_n^{\uparrow}$ and $p_n^{\downarrow}$ in the Corollary (3.6.2).

Corollary 3.6.2 *If* $p := \mathbb{P}(\{W_t\}_{\{q_1 \leq t \leq q\}} \in [W_s - \theta, W_s + \theta])$ *then its monotonically converging upper* $p_n^{\uparrow}$ *and lower* $p_n^{\downarrow}$ *sequence of probabilities can be constructed as*

$$p_n^{\downarrow} := \left(1 - \sum_{j=1}^{n} \zeta_{q-q_1}(j; W_{q_1} - W_q, \theta) + \sum_{j=1}^{n-1} \varphi_{q-q_1}(j; W_{q_1} - W_q, \theta)\right) \times \left(1 - \sum_{j=1}^{n} \zeta_{q_2-q}(j; W_q - W_{q_2}, \theta) + \sum_{j=1}^{n-1} \varphi_{q_2-q}(j; W_q - W_{q_2}, \theta)\right), \tag{3.6.4}$$

$$p_n^{\uparrow} := \left(1 - \sum_{j=1}^{n} \zeta_{q-q_1}(j; W_{q_1} - W_q, \theta) + \sum_{j=1}^{n} \varphi_{q-q_1}(j; W_{q_1} - W_q, \theta)\right) \times \left(1 - \sum_{j=1}^{n} \zeta_{q_2-q}(j; W_q - W_{q_2}, \theta) + \sum_{j=1}^{n} \varphi_{q_2-q}(j; W_q - W_{q_2}, \theta)\right), \tag{3.6.5}$$

with the property that

$$\lim_{n\to\infty} p_n^{\uparrow} \to p \quad \textit{and} \quad \lim_{n\to\infty} p_n^{\downarrow} \to p. \tag{3.6.6}$$

We use a similar argument presented in equations (3.5.10) and (3.5.11) to synthesize the Algorithm (3.6.1).

Algorithm 3.6.1: INTERMEDIATE BROWNIAN BRIDGE SIMULATOR$(q, q_1, W_{q_1}, q_2, W_{q_2}, W_s, \theta)$

output (W_q)
accepted $\leftarrow$ **false**
while accepted $\neq$ **true**
do $\begin{cases} W_q \sim N\left(W_{q_1} + \frac{(q-q_1)}{(q_2-q_1)}(W_{q_2} - W_{q_1}), \frac{(q-q_1)(q_2-q)}{(q_2-q_1)}\right) \\ U \sim U[0,1] \quad \text{and} \quad n = 1 \\ \textbf{while } p_n^{\downarrow} < U < p_n^{\uparrow} \text{ (as given in (3.6.4) and (3.6.5))} \\ \quad \textbf{do } n = n + 1 \\ \textbf{if } U \leq p_n^{\downarrow} \\ \quad \textbf{then accepted} \leftarrow \textbf{true} \end{cases}$
return (W_q)

3.7 Summary

This chapter introduced Brownian motion and methods to exactly simulate its sample trajectories. In particular, we discussed the methodology for the construction of Brownian motion sample paths using the localization method which is later used in Chapter (5) and (6). The localization method allows us to realize the sample path of Brownian motion under various restrictions which are of particular relevance in Chapter (5) and (6). For a user-specified hypercube, we simulated the sample path of a Brownian motion until its first passage time, which was discussed in Section (3.4). In such instances a user can fill the intermediate positions of Brownian motion using the simulation methodologies discussed in Section (3.5) and (3.6).

The localization method of simulating the exact sample paths of a Brownian motion is computationally intensive than the traditional method. This is mainly due to the fact that the localization method requires us to simulate the first passage times which is followed by the simulation of sample path conditioned to be restricted within a bound. This makes the localization method a computationally intensive simulation method. However, we should note that the use of the traditional method to simulate the sample paths of Brownian motion is not suitable in our case.

Chapter 4

Sampling from a Quasi-Stationary Distribution

In Chapter (1) we gave a motivation to the idea that simulation from a posterior distribution of interest can be achieved by suitably constructing a Markov process which exhibits a killing mechanism, whose quasi-stationary distribution can be used as a proxy for the target distribution. We will now review a number of different simulation mechanisms in order to simulate according to the quasi-stationary distribution of a Markov process. In particular, we focus on the regenerative approach proposed by [Blanchet et al., 2016] in order to simulate according to the quasi-stationary distribution of a Markov chain which exhibits absorbing states. This is a key constituent in the construction of the novel algorithm developed in this thesis. This regenerative approach simulates from the dynamics of the process until it gets absorbed. Upon absorption, it regenerates according to the empirical density of the states visited by the process. This method helps us in the construction of the novel algorithm in Chapter (6), where we obtain our sample from the quasi-stationary distribution of a *killed Brownian motion*.

This chapter is organised as follows: Section (4.1) outlines the concept of quasi-stationarity for a Markov process, followed by a heuristic argument to interpret the quasi-stationary distributions as the solution of an ordinary differential equation. This lays the motivation behind the construction of an algorithm to sample trajectories from the quasi-stationary distribution of a Markov process [Blanchet et al., 2016]. Furthermore, we present the iterative method by Blanchet et al. [2016] as a stochastic approximation scheme, but skip its proof in this thesis. Section (4.2) introduces other methods being used in the study of quasi-stationary behaviour of

a killed process. Finally Section (4.3) lays down a discussion on the strength and limitations of different approaches to quasi-stationarity.

4.1 Regenerative Approach for Simulating from a Quasi-Stationary Density

To understand the concept of the quasi-stationarity, we consider a Markov process $\{\mathbf{X}_t : t \geq 0\}$ which has absorbing states: if the process enters, it can never escape. The dynamics of such a process before it gets absorbed is termed as 'quasi-stationarity'.We begin this section by defining the concept of quasi-stationarity in the following sense:

Definition 4.1.1 ***Quasi-stationary:*** *Consider a Markov process $\{\mathbf{X}_t : t \geq 0\}$, which exhibits 'killing' (or visits absorbing states) where the killing time is denoted by a random variable ζ. A density π is called its quasi-stationary density if*

$$\mathbb{P}(\mathbf{X}_t \in \cdot \,|\, \zeta > t) = \pi(\cdot) \text{ for all } t > 0, \tag{4.1.1}$$

where $\mathbb{P}$ is the law of $\mathbf{X}_t$ conditional on the initial position $\mathbf{X}_0 \sim \pi$.

We define a related concept called quasi-limiting distributions.

Definition 4.1.2 ***Quasi-limiting:*** *Let π be a probability measure on the non-absorbing states of a Markov Chain $\{X_t : t \geq 0\}$ is called its quasi-limiting density if there exist a probability measure ν on the non-absorbing states such that for $\mathbf{X}_0 \sim \nu$ we have*

$$\lim_{t\to\infty} \mathbb{P}(\mathbf{X}_t \in \cdot \,|\, \zeta > t) = \pi(\cdot). \tag{4.1.2}$$

We should note that π is a quasi-limiting distribution of $\mathbf{X}_t$ if and only if it is a quasi-stationary distribution of $\mathbf{X}_t$ (see for instance [Méléard et al., 2012, Proposition 1]). If we assume the uniqueness of quasi-stationary density then one approach of finding a quasi-stationary density is to check if (4.1.2) converges to a finite limit and vice-versa. Hence, we would use the above two notions interchangeably.

The study of quasi-stationary distributions date back to the work of Yaglom [1947]. Later, the existence of a quasi-stationary distribution in the case of a Markov process on a discrete state-space was studied by Darroch and Seneta [1965]. The quasi-stationarity is an important element in the study of birth-death [Seneta and Vere-

Jones, 1966; van Doorn, 1991; Ferrari et al., 1991] and birth-catastrophe processes [Pakes and Pollett, 1989; Pakes, 1987; O'Neill, 2007]. Furthermore, it finds its use in a variety of applications including chemical kinetics [Dambrine and Moreau, 1981a,b], epidemics [Andersson and Britton, 2000; Nå sell, 1999], genetics [Ewens, 1963, 1964] and telecommunications [Pollett; Kelly, 1985] to name a few.

For the purpose of an illustration of the regenerative algorithm of [Blanchet et al., 2016], we restrict ourselves to a univariate continuous-time Markov chain $\{X_t : t \geq 0\}$ defined on a discrete state-space $\mathcal{S}$. Suppose d denotes the absorbing state and $\mathcal{T}$ is the set of all non-absorbing states. The quasi-stationary distribution of X_t will therefore be a distribution on non-absorbing states $\mathcal{T}$, conditioned on non-absorption of X_t until time t. We define the transition probabilities as

$$p_{0,t}(i,j) = \mathbb{P}(X_t = j | X_0 = i), \text{ and} \tag{4.1.3}$$

$$p_{0,t}(i,\mathcal{T}) = 1 - p_{0,t}(i,d). \tag{4.1.4}$$

Here $p_{0,t}(i,j)$ denotes the probability that the process transits to state j at time t starting from i at time 0. $p_{0,t}(i,\mathcal{T})$ denotes the probability that the process does not visit the absorbing state d starting from state i at time 0. As a result, the quasi-stationary density (under the assumption of unique quasi-stationary density) for the $j-th$ state of the Markov chain can be alternatively expressed as: (Darroch and Seneta [1967]; Zheng [2014])

$$\pi_j = \lim_{t\to\infty} \frac{p_{0,t}(i,j)}{p_{0,t}(i,\mathcal{T})}. \tag{4.1.5}$$

Recall that our aim is to simulate samples according to the quasi-stationary density $\pi(\cdot)$. We present the heuristic reasoning in the following section which can be interpreted to design the algorithm presented in [Blanchet et al., 2016].

4.1.1 An Alternative Representation of the Quasi-Stationary Density

In this section we outline an alternative representation of the quasi-stationary density for a continuous-time Markov chain defined on a discrete state space $\mathcal{S}$. This representation [de Oliveira and Dickamn, 2005] attains a differential equation form which can be interpreted to construct the regenerative algorithm outlined in [Blanchet et al., 2016]. To achieve this, we assume that $\mathcal{R}$ denotes the rate matrix of the continuous-time Markov process with absorbing state d. We observe using Kol-

mogorov's forward equation

$$\frac{d}{dt}(p_{0,t}(i,j)) = \sum_l p_{0,t}(i,l)\mathcal{R}(l,j), \tag{4.1.6}$$

where $\mathcal{R}(l,j)$ is the rate at which the process moves from state l to state j. Using (4.1.4) and (4.1.6) we have (ref to [de Oliveira and Dickamn, 2005; Blanchet et al., 2016] for conditions)

$$\frac{d}{dt}(p_{0,t}(i,\mathcal{T})) = \frac{d}{dt}(1 - p_{0,t}(i,d)) = -\frac{d}{dt}(p_{0,t}(i,d)) = -\sum_l p_{0,t}(i,l)\mathcal{R}(l,d). \tag{4.1.7}$$

The expression for the quasi-stationary distribution in (4.1.5) can attain the following form: ([de Oliveira and Dickamn, 2005; Blanchet et al., 2016])

$$\pi_j p_{0,t}(i,\mathcal{T}) \approx p_{0,t}(i,j) \quad \text{for large } t. \tag{4.1.8}$$

We should note that the form of a quasi-stationary distribution does not depend on t so differentiating (4.1.8) with respect to t yields

$$\pi_j \frac{d}{dt}(p_{0,t}(i,\mathcal{T})) \approx \frac{d}{dt}(p_{0,t}(i,j)) \tag{4.1.9}$$

$$= \sum_l p_{0,t}(i,l)\,\mathcal{R}(l,j). \quad \text{follows by (4.1.6)} \tag{4.1.10}$$

$$\approx \sum_l \pi_l\, p_{0,t}(i,\mathcal{T})\mathcal{R}(l,j), \quad \text{follows by (4.1.8)}. \tag{4.1.11}$$

Similarly, using (4.1.7) and (4.1.8) we have,

$$\frac{d}{dt}(p_{0,t}(i,\mathcal{T})) = -\sum_l p_{0,t}(i,l)\mathcal{R}(l,d) \tag{4.1.12}$$

$$\approx -\sum_l \pi_l\, p_{0,t}(i,\mathcal{T})\mathcal{R}(l,d). \tag{4.1.13}$$

Next, if we multiply (4.1.13) by π_j and subtract from (4.1.11) we have,

$$\sum_l \pi_l \, p_{0,t}(i,\mathcal{T})\mathcal{R}(l,j) + \pi_j \left(\sum_l \pi_l \, p_{0,t}(i,\mathcal{T})\mathcal{R}(l,d) \right) \approx \pi_j \frac{d}{dt}(p_{0,t}(i,\mathcal{T})) - \pi_j \frac{d}{dt}(p_{0,t}(i,\mathcal{T})) \tag{4.1.14}$$

$$\sum_l \pi_l \, p_{0,t}(i,\mathcal{T})\mathcal{R}(l,j) + \pi_j \left(\sum_l \pi_l \, p_{0,t}(i,\mathcal{T})\mathcal{R}(l,d) \right) \approx 0 \tag{4.1.15}$$

$$\left(\sum_l \pi_l \, \mathcal{R}(l,j) + \pi_j \left(\sum_l \pi_l \, \mathcal{R}(l,d) \right) \right) p_{0,t}(i,\mathcal{T}) \approx 0 \tag{4.1.16}$$

$$\sum_l \pi_l \, \mathcal{R}(l,j) + \pi_j \left(\sum_l \pi_l \, \mathcal{R}(l,d) \right) \approx 0. \tag{4.1.17}$$

We note that the quasi-stationary density of the $j-th$ state is independent of time. Therefore, (4.1.17) can be rewritten in the differential form as:

$$\sum_l \pi_l \, \mathcal{R}(l,j) + \pi_j \left(\sum_l \pi_l \, \mathcal{R}(l,d) \right) \approx \frac{d}{dt}(\pi_j). \tag{4.1.18}$$

Therefore, π can be interpreted as the stationary point of the forward equation (4.1.18). The first part in the LHS of (4.1.18) is the term from the general Kolmogorov forward equation. In the second part, $\left(\sum_l \pi_l \, \mathcal{R}(l,d) \right)$ is the probability of hitting the absorbing state starting from a non-absorbing state distributed according to π. We can use this idea in algorithm (4.1.1) (Blanchet et al. [2016]). We simulate the chain starting from a non-absorbing state until it hits the absorbing state. Once it hits the absorbing state, we simulate the starting position based on the empirical density of non-absorbing states traced by the chain. As $t \to \infty$, samples drawn are from the quasi-stationary distribution (Blanchet et al. [2016]).

Algorithm 4.1.1: SIMULATING FROM A QUASI STATIONARY DENSITY(π_0, t)

output ($\mathbf{S}$: Skeleton of a given chain)

1. Initialize the probability vector $\tilde{\pi} = \pi_0$ on the non-absorbing states of Markov chain.
2. Set $s = 0$, simulate $X_s \sim \tilde{\pi}$ and set $\mathbf{S} \leftarrow \mathbf{S} \cup X_s$.

while $s < t$

do
- **while** Segment is not absorbed
  - **do**
    3. Set $\tilde{s} = s + ds$,simulate $X_{\tilde{s}}$ according to the law of $X_{\tilde{s}} \,|\, X_s$.
    4. Update $\mathbf{S} \leftarrow \mathbf{S} \cup X_{s+ds}$.
    5. Update $\tilde{\pi}$ using $\tilde{\pi}_j = \frac{1}{\tilde{s}} \int_0^{\tilde{s}} \mathbb{I}(X_q = j) dq$ for all $j \in \mathcal{T}$

    until absorption.
- 6. Once absorbed at $\tilde{s}$, re-simulate $X_{\tilde{s}} \sim \tilde{\pi}$ and set $s = \tilde{s}$.

return ($\mathbf{S}$)

We should note that Algorithm (4.1.1) simulates the trajectory of a killed Markov chain at equally spaced time point of length ds. Here we should note that the time-discretization has been chosen for illustration purposes only, without influencing the exactness of Algorithm (4.1.1). Similarly a user-specified sequence of time points can be chosen to obtain a trajectory of a killed Markov process until absorption which is followed by the regeneration at the instance of kill and so on.

Subsection (4.1.2) discusses the outline of the proof of Algorithm (4.1.1) for a discrete-time absorbing Markov chain defined on a discrete state-space. The proof uses a stochastic approximation method which converges to the quasi-stationary distribution of the Markov chain [Kushner and Yin, 2003, Chapter 5].

4.1.2 Validity of the Regenerative Algorithm (4.1.1)

In this section we outline the proof of Algorithm (4.1.1) for the discrete-time absorbing Markov chain defined on a discrete state-space, by representing the regenerative algorithm (4.1.1) as a recursive stochastic approximation update. Once framed as a stochastic approximation update, this allows us to obtain its long term behaviour using the solutions of a suitably constructed differential equation [Kushner and Yin, 2003, Theorem 5.2.1]. In a discrete state-space setting, these solutions can be viewed as the quasi-stationary density of the underlying process (as obtained by Algorithm

(4.1.1)). A formal extension of algorithm (4.1.1) to 'discrete-time general state-space Markov chain' and 'continuous-time discrete state-space Markov chain' can be found in Zheng [2014]; their discussion is beyond the scope of this thesis. However, no such extensions exists for a continuous-time absorbing Markov chain defined on a general state-space.

Recursive Stochastic Approximation Algorithm

The basic stochastic approximation can be represented in the recursive form as ([Robbins and Monro, 1951], [Kushner and Yin, 2003, Chapter 1])

$$\theta_{n+1} = \theta_n + \epsilon_n \mathbf{Y}_n, \text{ for } \theta_n, \mathbf{Y}_n \in \mathbb{R}^p, \tag{4.1.19}$$

where θ_0 is given and the noise sequence $\{\epsilon_n\}_{n\geq 1}$ is deterministic with the property that

$$\epsilon_n \to \infty, \sum_{n=1}^{\infty} \epsilon_n = \infty \text{ and } \sum_{n=1}^{\infty} \epsilon_n^2 < \infty. \tag{4.1.20}$$

Furthermore, $\{\mathbf{Y}_n\}_{n\geq 1}$ is an observed sequence of random variables measurable with respect to $\mathcal{F}_n := \sigma\{(\theta_i, \mathbf{Y}_{i-1}) : 1 \leq i \leq n\}$ with the property that $\mathbb{E}(\mathbf{Y}_n \,|\, \mathcal{F}_n) = g(\theta_n)$, for some 'nice' function $g(\cdot)$ (see [Kushner and Yin, 2003, Chapter 4,5] for more details). Under some regularity conditions (see [Kushner and Yin, 2003, Chapter 5.2.1] and [Blanchet et al., 2016] for more details) it can be shown that θ_n converges to a stable attractor of a differential equation of the form:

$$\frac{d}{dt}(\theta(t)) = g(\theta(t)). \tag{4.1.21}$$

Algorithm (4.1.1) Represented as a Stochastic Approximation Algorithm

Let $\mathcal{S}$ be the state-space of the Markov chain with $\mathcal{T} \subset \mathcal{S}$ being the set of transient states. Further, let Q be the sub-stochastic matrix over the set of transient states $\mathcal{T}$ with π_n as the sequence of a normalized probability vector over $\mathcal{T}$ after the n^{th} iteration of Algorithm (4.1.1). Thus $\pi_n(x)$ stores the cumulative empirical distribution up to and including the n^{th} iteration of Algorithm (4.1.1) for the transient state x. We define $\{X_k^n\}$ as the Markov chain used in the n^{th} iteration of the Algorithm (4.1.1). Next, we define the hitting time of the absorbing states in the n^{th} iteration of the Markov chain $\{X_k^n\}_{k\geq 1}$ as $\tau^{(n)} = \min\{k \geq 0 : X_k^n \notin \mathcal{T}\}$. The iterative updating of π in step 5 of Algorithm (4.1.1) is then as follows:

$$\pi_{n+1}(j) = \frac{\left(\sum_{k=0}^{n} \tau^{(k)}\right) \pi_n(j) + \sum_{k=0}^{\tau^{(n+1)}-1} \mathbb{I}(X_k^{(n+1)} = j | X_0^{(n+1)} \sim \pi_n)}{\sum_{k=0}^{n+1} \tau^{(k)}}. \tag{4.1.22}$$

Equation (4.1.22) can be further reduced to

$$\pi_{n+1}(j) = \pi_n(j) + \frac{\sum_{k=0}^{\tau^{(n+1)}-1} \mathbb{I}(X_k^{(n+1)} = j | X_0^{(n+1)} \sim \pi_n) - \tau^{(n+1)} \pi_n(j)}{\sum_{k=0}^{n+1} \tau^{(k)}}. \tag{4.1.23}$$

The iterative procedure (4.1.23) is expressed as a stochastic approximation algorithm similar to (4.1.19):

$$\pi_{n+1}(j) = \pi_n(j) + \frac{1}{n+1} \frac{\sum_{k=0}^{\tau^{(n+1)}-1} \left(\mathbb{I}(X_k^{(n+1)} = j | X_0^{(n+1)} \sim \pi_n) - \pi_n(j)\right)}{\frac{1}{n+1} \sum_{k=0}^{n+1} \tau^{(k)}}. \tag{4.1.24}$$

The sequence (π_n) takes values in a probability simplex. The measurability assumption in the stochastic approximation algorithm (4.1.19) requires that the second summand in (4.1.24) depends only on π_n. However, the denominator $\sum_{k=0}^{n+1} \tau^{(k)}$ in (4.1.24) depends on the whole history of sequence π_n. To circumvent this issue, Blanchet et al. [2016] added another iterative state T_n by defining

$$T_n = \frac{1}{n+1} \sum_{k=0}^{n} \tau^{(k)}, \tag{4.1.25}$$

which iterated as

$$T_{n+1} = T_n + \frac{1}{n+2}(\tau^{(n+1)} - T_n). \tag{4.1.26}$$

Now (4.1.24) can be rewritten as

$$\pi_{n+1}(j) = \pi_n(j) + \frac{1}{n+2}\left(\frac{n+2}{n+1}\right)\frac{\sum_{k=0}^{\tau^{(n+1)}-1}\left(\mathbb{I}(X_k^{(n+1)} = j | X_0^{(n+1)} \sim \pi_n) - \pi_n(j)\right)}{T_n + \frac{\tau^{(n+1)}}{n+1}}. \tag{4.1.27}$$

The term $\frac{\tau^{(n+1)}}{n+1}$ in the denominator is asymptotically negligible. We define

$$\mathbf{Y}_n^1(\pi^\top, T)(j) := \mathbf{Y}_n^1(\pi(j), T) := \left(\frac{n+2}{n+1}\right)\frac{\sum_{k=0}^{\tau-1}\left(\mathbb{I}(X_k = j | X_0 \sim \pi) - \pi(j)\right)}{T + \frac{\tau}{n+1}} \text{ and} \tag{4.1.28}$$

$$\mathbf{Y}_n^2(\pi^\top, T) := (\tau - T). \tag{4.1.29}$$

Further, we define $\mathbf{Y}_n(\pi^\top, T) = (\mathbf{Y}_n^1(\pi^\top, T), \mathbf{Y}_n^2(\pi^\top, T))'$ where $\mathbf{Y}_n^1(\pi^\top, T) \in \mathbb{R}^r$ is the vector consisting of $\mathbf{Y}_n^1(\pi^\top, T)(j_i)$ and $\{j_1, ..., j_r\}$ is the set of the transient states. Thus,

$$\begin{pmatrix}\pi_{n+1}\\ T_{n+1}\end{pmatrix} = \begin{pmatrix}\pi_n\\ T_n\end{pmatrix} + \epsilon_n \times \begin{pmatrix}\mathbf{Y}_n^1(\pi^\top, T)\\ \mathbf{Y}_n^2(\pi^\top, T)\end{pmatrix}, \quad \text{where} \quad \epsilon_n = \frac{1}{n+2}. \tag{4.1.30}$$

We are now able to express (4.1.30) in the form of (4.1.19) :

$$\theta_n = \begin{pmatrix}\pi_n\\ \mathbf{T}_n\end{pmatrix}, \quad \mathbf{Y}_n = \begin{pmatrix}\mathbf{Y}_n^1(\pi^\top, T)\\ \mathbf{Y}_n^2(\pi^\top, T)\end{pmatrix}, \tag{4.1.31}$$

where $\epsilon_n = \frac{1}{n+2}$ clearly satisfies (4.1.20). Here θ_n lies in $H = H^1 \times [0, \infty)$ where $H^1 = \{\mathbf{j} = (j_1, ..., j_r) \in \mathbb{R}_+^r | \sum_{i=1}^r \pi_n(j_i) = 1\}$ for each n. Building on the foundations of the stochastic approximation algorithm given in [Kushner and Yin, 2003, Theorem 5.2.1], Blanchet et al. [2016] proved the convergence result. Here, we state the result by Blanchet et al. [2016].

Theorem 4.1.1 ([Blanchet et al., 2016]) *Given an irreducible absorbing Markov chain over a finite state space $\mathcal{S}$, with Q as the sub-stochastic matrix over the transient states, let π_0 be an initially chosen probability vector over the non-absorbing states and $T_0 \geq 1$ be the initial value of T_n (T_n as defined in (4.1.25)). Then, there*

exists a unique quasi-stationary distribution π *such that*

$$\pi^\top Q = \rho\pi^\top \text{ with } \pi = (\pi(j_1), ..., \pi(j_r))^\top \geq 0, \sum_{i=1}^{r} \pi(j_i) = 1. \tag{4.1.32}$$

Here ρ *is an eigenvalue corresponding to eigenvector* π*. Therefore, the algorithm (4.1.1) with iterates* $\theta_n = (\pi_n^\top, T_n)$ *and initial values* $(\pi_0^\top, T_0)$ *converges with probability one to the point* $(\pi^\top, 1/(1-\lambda))$ *where* λ *is the principle eigenvalue of* Q*. Furthermore, if* $\bar{\lambda}$ *is any non-principal eigenvalue of* Q *with* $\bar{\lambda} \neq \lambda$ *and*

$$Re\left(\frac{1}{1-\bar{\lambda}}\right) < \frac{1}{2}\left(\frac{1}{1-\lambda}\right), \tag{4.1.33}$$

then $\sqrt{n}(\pi_n - \pi)$ *converges in distribution to a* $N(0, \Sigma_0)$ *for some* Σ_0*.*

A similar result holds for the continuous-time absorbing Markov chain on a discrete state-space and discrete-time absorbing Markov chain on a general state-space [Zheng, 2014]. Mailler and Villemonais [2018] have demonstrated its convergence on compact and non-compact spaces under certain conditions.

Mailler and Villemonais [2018] extend the stochastic approximation algorithm for quasi-stationarity to killed Markov processes evolving in non-compact state spaces. They establish almost-sure convergence of the stochastic approximation algorithm defined on the space of measures over a non-compact space. However, Mailler and Villemonais [2018] assume (assumption under [Mailler and Villemonais, 2018, Proposition 5]) that the the killed diffusion process under consideration has a linear inward drift. This does not allow the process to stay in the tails of the quasi-stationary distribution for long and is constantly pushed towards the centre of the distribution. As a result it can be inefficient while simulating from a heavy-tailed quasi-stationary distribution since the algorithm would not explore tails properly.

4.2 Alternative Methods to Simulate from a Quasi-Stationary Density

4.2.1 Sequential Monte Carlo Method

Sequential Monte Carlo (SMC) provides a general framework to produce a weighted sample from a sequence of target measures $\{\pi_t(\mathbf{x}_{0:t})\}$ (where $\mathbf{x}_{0:t} = \mathbf{x}_0, ..., \mathbf{x}_t$), where each density is defined on some sequence of increasing product space E^t [Gordon et al., 1993; Doucet et al., 2000; Del Moral et al., 2006; Doucet and Johansen,

2011; Del Moral et al., 2011]. These sequence of densities are typically known up to a normalising constant Z_t i.e. $\pi_t = \gamma_t / Z_t$. This is often used for estimating unknown quantities where observations arrive sequentially in time. Therefore, a user performs an on-line inference by updating the posterior distribution based on the incoming data. SMC finds its use in a wide variety of applications including target tracking [Gordon et al., 1993; Hue et al., 2002; Frank et al., 2003], real-time enhancement of audio signals [Godsill et al., 2002; Fong et al., 2002; Vermaak et al., 2002] and image sequence tracking [Brasnett et al., 2007; Arnaud and Mémin, 2007] to name a few. A recent application involved simulating according to the quasi-stationary density of a Markov process, which exhibited a killing mechanism [Pollock et al., 2017]. This application is of particular relevance in our context.

An SMC sampler often uses an importance sampling algorithm (see Section (2.2)) to sequentially propose samples for the target density of interest. Recall that, within an importance sampling framework samples are proposed according to a density $q(\cdot)$ for the target density $\pi(\cdot)$ [Robert and Casella, 2013, Section 3.3]. An importance sampler outputs proposed samples $\{X_i : i = 1, ..., N\}$ together with their importance weights $\{w^*(X_i) := \pi(X_i)/q(X_i) : i = 1, ..., N\}$ which can be suitably combined to estimate the target density. In this setting, a proposal density is chosen such that it minimizes the variance of the scaled samples $\{X_i \times w^*(X_i) : i = 1, ..., N\}$ [Robert and Casella, 2013; Geweke, 1989].

Within the SMC framework, we choose a sequence of proposal densities $\{q_t(\mathbf{x}_{0:t})\}$ in order to draw proposed samples and compute importance weights defined on the product space E^t. In our context a choice of the proposal density is made such that

$$q_t(\mathbf{x}_{0:t}) = q_0(\mathbf{x}_0) \times \prod_{i=1}^{t} q_i(\mathbf{x}_i \,|\, \mathbf{x}_{i-1}), \tag{4.2.1}$$

which also reduces the computational cost [Doucet and Johansen, 2011]. This helps us to draw samples according to q_t by simply simulating according to the sequence of densities $q_0(\mathbf{x}_0), q_1(\mathbf{x}_1 \,|\, \mathbf{x}_0), ..., q_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})$. This gives us the following form of

the importance weights:

$$w_t^*(\mathbf{x}_{1:t}) = \frac{\gamma_t(\mathbf{x}_{1:t})}{q_t(\mathbf{x}_{1:t})} \tag{4.2.2}$$

$$= \frac{\gamma_{t-1}(\mathbf{x}_{1:t-1})}{q_0(\mathbf{x}_0) \times \prod_{i=1}^{t-1} q_i(\mathbf{x}_i \,|\, \mathbf{x}_{i-1})} \times \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1}) \times q_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})} \tag{4.2.3}$$

$$= w_{t-1}^*(\mathbf{x}_{1:t-1}) \times \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1}) \times q_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})} \tag{4.2.4}$$

$$:= w_{t-1}^*(\mathbf{x}_{1:t-1}) \times \tilde{w}_{t-(t-1)}^*(\mathbf{x}_{1:t}). \tag{4.2.5}$$

This provides a mechanism to compute the importance weight iteratively using the incremental weight $\tilde{w}^*$. If we are interested in simulating N samples according to $\pi_t(\mathbf{x}_{0:t})$, we can propagate N particles and renormalise weights at each step using

$$w_t^{(i)} = w_t^*(\mathbf{x}_{0:t}^{(i)}) / \sum_{j=1}^{N} w_t^*(\mathbf{x}_{0:t}^{(j)}), \tag{4.2.6}$$

which can be used to estimate the target density:

$$\pi_t(\mathbf{x}_{0:t}) \approx \tilde{\pi}_t(\mathbf{x}_{0:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}). \tag{4.2.7}$$

However, the above outlined procedure suffers from particle weight degeneracy for large values of t [Collet et al., 2012]. In simple terms, one particle might carry a large weight after normalisation (4.2.6), while other particles at the same time, carrying extremely low weights. The particle re-sampling techniques are usually used to address this problem [Gordon et al., 1993]. This can be implemented by simply fixing a sequence of re-sampling time points, say $t_1, ..., t_m$, where we re-sample particles according to the empirical distribution of their normalised weights [Doucet and Johansen, 2011].

As noted above, re-sampling removes particles with low weights and duplicates particles with high weights, which introduce some additional variances in the weights of particles. If the unnormalised weights of particles are small the re-sampling step can be unnecessary. Therefore, at each re-sampling time point, it is reasonable to re-sample if the variance of the unnormalised weights is greater than a user-specified threshold. This variability is often measured using the effective sample size, $N_{eff} := 1/\sum_{i=1}^{N} \left(w_t^{(i)}\right)^2$ and the re-sampling is carried out if the effective sample

size of the collection of weights N_{eff} falls below a user-specified threshold, say N_{th} [Liu, 2008; Doucet and Johansen, 2011]. We present a simplified version of the above procedure where the re-sampling step is performed at the time of simulation if the effective sample size falls below the threshold specified by the user. This procedure has been synthesized in Algorithm (4.2.1).

Algorithm 4.2.1: SEQUENTIAL IMPORTANCE RESAMPLING(N, q, N_{th})

Input : N : number of particles, q : proposal density

N_{th} : Effective sample size threshold

output ($\mathbf{S}$: Positions along with their weights)

for each $i \in \{1, ..., N\}$

do { sample $\mathbf{X}_0^{(i)} \sim q_0$ and set $w_0^{(i)} \leftarrow 1/N$
calculate $\tilde{\pi}_t(\mathbf{x}_0) = \sum_{i=1}^{N} w_0^{(i)}(\mathbf{x}_0)\delta_{\mathbf{x}_0^{(i)}}(\mathbf{x}_0)$

for each $t > 0$

do {
calculate $N_{eff} = 1/\sum_{i=1}^{N}\left(w_t^{(i)}\right)^2$
for each $i \in \{1, ..., N\}$
do { **if** $N_{eff} < N_{th}$
then { **resample** $\mathbf{X}_{0:t-1}^{(i)} \sim \tilde{\pi}_{t-1}^{N}$
set $w_{t-1}^{(i)} = 1/N$ for each resampled particle
for each $i \in \{1, ..., N\}$
do { **simulate** $\mathbf{X}_t^{(i)} \sim q_t\left(\cdot \,\middle|\, \mathbf{X}_{t-1}^{(i)}\right)$ and the weight $\tilde{w}_{t-(t-1)}^{*(i)}$ as per eq (4.2.4).
update $w_t^{*(i)} = w_{t-1}^{*(i)} \times \tilde{w}_{t-(t-1)}^{*(i)}$
for each $i \in \{1, ..., N\}$
do normalise $w_t^{(i)} = w_t^{*(i)} / \sum_{k=1}^{N} w_t^{*(k)}$
$\mathbf{S} \leftarrow \mathbf{S} \cup (\mathbf{X}_t^{(1:N)}, w_t^{(1:N)})$
set $\tilde{\pi}_t(\mathbf{x}_{0:t}) = \sum_{i=1}^{N} w_t^{(i)}(\mathbf{x}_{0:t})\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})$.

return ($\mathbf{S}$)

Finally, Algorithm (4.2.1) can be used to estimate the quasi-stationary density of interest. At any given instance t, a proposal is made according to a suitably chosen q_t for $\mathbb{P}(\mathbf{X}_t \in \cdot \,|\, \zeta > t)$ and its survival probability until time t is chosen as the importance weight within Algorithm (4.2.1) [Pollock et al., 2017]. After a sufficiently long

time t, the estimate given in (4.2.7) can be used as a proxy for the quasi-stationary density.

4.2.2 The Fleming–Viot Method

In this section, we briefly layout the Fleming-Viot system for sampling according to the quasi-stationary density of a killed process [Fleming and Viot, 1979]. The Fleming–Viot type system and its variants have appeared in the study of the quasi-stationarity of an integrating particle system [Ferrari and Marić, 2007; Bieniek et al., 2012; Asselah et al., 2011; Grigorescu and Kang, 2004; Groisman and Jonckheere, 2013]. For a killed Brownian motion, the Fleming–Viot type system was introduced by Burdzy et al. [1996] and later explored in [Burdzy et al., 2000; Grigorescu and Kang, 2004]. It was later studied for a multi-dimensional diffusion with unbounded drifts in [Villemonais, 2011a,b].

Consider a killed Markov process $\{\mathbf{X}_{0:t}\}$ which evolves in its state space E. In this method, we fix $N \geq 2$ particles, initialise them at $(\mathbf{X}_0^1, ..., \mathbf{X}_0^N)$ and allow them to independently evolve according to the dynamics of the given killed process $\mathbf{X}_t$ until one of the particles is absorbed. Once a particle is absorbed, its position is *revived* by jumping to one of the positions of the remaining $N-1$ particles. This position is chosen uniformly at random [Burdzy et al., 2000; Méléard et al., 2012; Oçafrain and Villemonais, 2017]. Once re-sampled, particles evolve independently according to the dynamics of the killed Markov process until absorption and so on. As the number of particles $N \to \infty$, the empirical distribution of the surviving particles at a fixed time converges to the quasi-stationary distribution of the underlying dynamics. This allows us to sample according to the quasi-stationary density of a Markov process defined on a countable state-space, where the process is likely to be absorbed. We should note that a Fleming–Viot method can be viewed as a pure rebirth version of an SMC-based approach.

In a Fleming–Viot type framework, a user can find herself in a situation where all particles get absorbed simultaneously. One cannot revive any particle and hence this process is stopped. However, this situation is unlikely to arise when the number of particles is kept large, since the probability of all of them being simultaneously absorbed is negligible. Moreover, an N–particle Fleming–Viot system for a continuous-time process often requires that only one particle gets absorbed at an instance [Burdzy et al., 1996]. This can be problematic when two or more particles are absorbed simultaneously. Oçafrain and Villemonais [2017] proposed a different

system of interacting particles which circumvents this problem. They consider a discrete time N–particle system $(\mathbf{X}_t^1, ..., \mathbf{X}_t^N)$ initialised at $(\mathbf{x}_t^1, ..., \mathbf{x}_t^N)$ indexed by $(b_1, ..., b_N) \in \{0, 1\}^N$. The dynamics between t and $t+1$ is given as follows: at time t each particle is indexed by setting $(b_1, ..., b_N) = (0, ..., 0)$. Then a particle (say i^{th}) is chosen uniformly which also satisfies $b_i = 0$. A position, say Z, is then simulated according to $\mathbb{P}(\mathbf{X}_{t+1} \in \cdot \,|\, \mathbf{X}_t = \mathbf{x}_t^i)$. If the i^{th} particle is absorbed at Z then its position and index are revived by uniformly choosing among $N-1$ remaining particles, i.e. $(\mathbf{x}_t^i, b_i) = (\mathbf{x}_t^j, b_j)$ where $j \sim U\{1, .., N\} \setminus \{i\}$. Otherwise, its position and index are replaced by Z and 1 respectively. The combination of the above steps are repeated until $b_i = 1$ for each $i \in \{1, .., N\}$. Then the position at time $t+1$ is set as $(\mathbf{x}_t^1, ..., \mathbf{x}_t^N)$. Oçafrain and Villemonais [2017] show that the above interacting particle system follows the same law as a suitably constructed Fleming–Viot system for a continuous-time process; this converges to the required quasi-stationary density [Oçafrain and Villemonais, 2017].

4.3 A Comparison of Three Approaches Considered

We should note that the regenerative approach [Blanchet et al., 2016] consider the evolution of a single trajectory over a sufficiently long time t, however, a particle-based system (namely SMC and Fleming–Viot system) relies on a large number of particles N for its convergence. From an algorithmic point of view, the cost of evolution at any instance in the regenerative approach of [Blanchet et al., 2016] is $\mathcal{O}(N)$ times cheaper than a particle-based method. At the same time, the computational cost involved in the regeneration step in Algorithm (4.1.1) is a function of t, whereas, the re-sampling step in a particle-based method is a function of the number of particles N. Therefore, the algorithmic cost of regeneration is expected to grow as a function of time t in Algorithm (4.1.1), while the re-sampling cost remains constant ($\mathcal{O}(N)$) for a fixed number of particles. As far as the implementation is concerned, the regenerative approach (4.1.1) is easier to implement as compared to a particle-based approach considered here. This is mainly due to the fact that maintaining the time evolution of a computational object for N particles is far more challenging in the case of a particle-based method.

In a Fleming–Viot type system, the number of particles remaining alive decrease at an exponential rate as a function of time and are actually used to estimate the quasi-stationary density. Precisely, for a finite number of particles N, the number of surviving particles vanishes with probability one in a finite time t [Oçafrain and

Villemonais, 2017]. Therefore, a Fleming–Viot system is usually constructed upon a large number of particles in order to achieve a desired convergence. As noted before, this leads to a huge computational bottleneck in the implementation of this method. However, the choice of the suitable values of N and t can often be challenging in a particle-based method. In such situations, it is often unclear how large/small the values of N and t should be chosen with respect to one another. Therefore, the number of particles N is usually chosen in an ad hoc manner. To address this problem, Elvira et al. [2017] introduced an adaptive selection of the number of particles in an SMC-based method. Despite this a suitable choice of t still remains an issue. Furthermore, a particle-based system often faces the curse of dimensionality. To put formally, its computational cost increases exponentially with the number of dimensions [Bengtsson et al., 2008]. However, the computational cost with respect to the dimensions of the regenerative mechanism have not been established in [Blanchet et al., 2016]; we discuss more on its behaviour in Chapter (6).

On the flip side, the regenerative algorithm (4.1.1) maintains a 'strong memory' of its path visited in the past. This can be problematic in a situation when the algorithm is initialised poorly or experiences a bad start. In this case, the algorithm can continue to explore the non-interesting regions of the underlying quasi-stationary density, which ultimately results in a poor regeneration as well. Therefore, the algorithm might lead to poor convergence or no convergence at all. From a theoretical point of view, there exists a number of literatures supporting the convergence of SMC-based methods and the Fleming-Viot approach. In particular, there exists a sound mathematical foundation and the rate of convergence for Algorithm (4.2.1). On the contrary regenerative Algorithm (4.1.1) lacks the mathematical building block in a more general setting. On the positive side, this work also provides an empirical support to the validity of Algorithm (4.1.1) in a continuous state-space setting which has not been studied before.

Keeping in mind that a poor initialisation can often become problematic in the regenerative Algorithm (4.1.1), we propose a new class of regeneration strategies in Chapter (7) and embed it within our setting. We empirically validate the working of such regeneration strategies when the resulting algorithm is subjected to a poor start (see Chapter (7)). However, we should note that the mathematical foundations of this approach have not been established in this thesis.

Chapter 5

The Construction of a QSMC Method

This chapter provides the necessary mathematical building block which is being used in the construction of the algorithm developed in this thesis. It provides a new mathematical framework known as the *Quasi-Stationary Monte Carlo (QSMC)* which can be used to make exact draws from an intractable distribution of interest. As noted in Chapter (1), the QSMC method utilises the quasi-stationary density of a Markov process killed at a suitably constructed hazard rate, which acts as a proxy for the target density. In a tall data setting, the mathematical form of this hazard rate allows us to employ the sub-sampling method in order to compute its unbiased estimator. When the sub-sampling idea is used within the QSMC method, it provides a sub-linear scalability to the resulting *Scalable Langevin Exact (ScaLE)* algorithm.

This chapter is organised as follows: Section (5.1) lays down the intuitive reasoning behind the construction of a QSMC method. It illustrates that a problem of sampling from an intractable distribution can be viewed as drawing samples from the quasi-stationary distribution of a suitably constructed killed Markov process. Section (5.2) provides the algorithmic construction which employs the ideas from Section (5.1) in order to obtain samples from a given target distribution. Section (5.3) provides a crucial redesign of the algorithm presented in Section (5.2) which gives the algorithm a sub-linear scalability. It employs the sub-sampling method together with the control variate construction in order to achieve the scalability with respect to the data size.

5.1 QSMC : Heuristics and Construction

We lay down the foundation of the QSMC approach in order to obtain samples from our target density π. We motivate the mathematical construction of QSMC using the properties of a Langevin diffusion (5.1.1). However, we should note that the Langevin dynamics is not being directly used in the construction of the algorithm developed in this chapter. As observed in Chapter (1), we can draw samples from a target density π by following the dynamics of a Langevin diffusion

$$d\mathbf{X}_t = \frac{1}{2}\nabla \log(\pi(\mathbf{X}_t))dt + d\mathbf{W}_t, \quad \mathbf{X}_0 = \mathbf{x}_0 \in \mathbb{R}^d, t \in [0, \infty), \tag{5.1.1}$$

whose stationary density coincides with π. The *occupation measure* of the trajectories of this Langevin diffusion defined by

$$\pi_t(B) := \frac{1}{t}\int_0^t \mathbb{I}_B(\mathbf{X}_s)ds, \quad \text{for } B \subset \mathbb{R}^d \tag{5.1.2}$$

can be treated as a proxy for π, after a sufficiently long time $t > 0$. However, the storage of a continuous-time object over a long time horizon is impossible. Therefore, we can simulate and store a finite-dimensional representation of the trajectories (called skeleton) of this diffusion. In order to simulate the trajectories of the above Langevin dynamics, we need to be able to simulate according to the transition density, say $p_t(\mathbf{x}_0, \mathbf{x}_t)$ of (5.1.1). Denoting the drift component by $\beta(\mathbf{x}) := \frac{1}{2}\nabla \log(\pi(\mathbf{x}))$ and using the Dacunha-Castelle formula [Dacunha-Castelle and Florens-Zmirou, 1986], we get the following form of the transition density:

$$p_t(\mathbf{x}_0, \mathbf{x}_t) \propto \exp\left(-\frac{||\mathbf{x}_t - \mathbf{x}_0||^2}{2t}\right)(\pi(\mathbf{x}_t))^{1/2} \times \mathbb{E}_{\mathbb{W}}\left(\exp\left\{-\int_0^t \phi(\mathbf{X}_s)ds\right\}\middle| \mathbf{X}_0 = \mathbf{x}_0, \mathbf{X}_t = \mathbf{x}_t\right). \tag{5.1.3}$$

Here $\phi(\mathbf{X}_s) := \left(||\beta(\mathbf{X}_s)||^2 + \text{Div}\beta(\mathbf{X}_s)\right)/2$, $\mathbb{E}_{\mathbb{W}}$ denotes the expectation with respect to a Brownian bridge, $||\cdot||$ is the Euclidean norm and Div is a divergence operator. For $\phi(\cdot)$ to be well defined, we require the following condition:

Condition–0 : The gradient $\nabla \log(\pi(\mathbf{x}))$ and divergence $\text{Div}\nabla \log(\pi(\mathbf{x}))$ exists for all $\mathbf{x}$.

In this case the transition density $p_t(\cdot,\cdot)$ converges to π as t approaches infinity. Therefore, if it were possible, we could simulate the trajectories of the Langevin diffusion (5.1.1) using our transition density $p_t(\cdot,\cdot)$ and treat the occupation measure as the proxy for the target density π.

An exact approach to simulate according to the transition density $p_t(\cdot,\cdot)$ is to employ a rejection sampler wherein we simulate the terminal position $\mathbf{X}_t$ according to a proposal

$$q(\mathbf{x}_t) \propto \exp\left(-\frac{||\mathbf{x}_t - \mathbf{x}_0||^2}{2t}\right)(\pi(\mathbf{x}_t))^{1/2}, \tag{5.1.4}$$

and accept or reject the entire trajectory $\{\mathbf{X}_s \,|\, 0 \leq s \leq t\}$ with a probability proportional to $\mathbb{E}_{\mathbb{W}}\left(\exp\left\{-\int_0^t \phi(\mathbf{X}_s)ds\right\}\middle|\mathbf{X}_0 = \mathbf{x}_0, \mathbf{X}_t = \mathbf{x}_t\right)$ [Beskos et al., 2006, 2008]. However, this approach carries two-fold limitations. First, the problem of simulating the terminal position as per (5.1.4) is as hard as the original problem itself. This is due to the presence of term $(\pi(\mathbf{x}))^{1/2}$ in (5.1.4) and hence its implementation is highly impractical. Secondly, the acceptance probability diminishes at an exponential rate which can further gravitate the computational inefficiencies in the acceptance of a proposed terminal position.

In an ideal scenario, if the middle term $(\pi(\mathbf{x}))^{1/2}$ in the transition density (5.1.3) is missing, the implementation of the aforementioned approach is possible. This is equivalent to a situation where we follow the dynamics having the transition density $p_t(\mathbf{x}_0,\mathbf{x}) \times \pi(\mathbf{x})^{-1/2}$ [Pollock et al., 2017]. However, in this approach we would bias the final invariance of the original Langevin diffusion by a factor of $(\pi(\mathbf{x}))^{1/2}$ and hence the transition density would converge to $(\pi(\mathbf{x}))^{1/2}$ (intuitively $\lim_{t\to\infty} p_t(\mathbf{x}_0,\mathbf{x}) \times \pi(\mathbf{x})^{-1/2} \to \pi(\mathbf{x})^{1/2}$). Therefore, in order to achieve the correct target density one could consider a Langevin diffusion with doubled drift i.e. $\beta(\mathbf{x}) := \nabla \log(\pi(\mathbf{x}))$, for which the invariant density is given by π^2. In this situation, the transition density (5.1.3) (when $\beta(\mathbf{x}) := \nabla \log(\pi(\mathbf{x}))$ is used) would have a factor of $\pi(\mathbf{x})$ in the middle term. Hence, its exclusion allows us to retrieve our correct target density. For $\beta(\mathbf{x}) := \nabla \log(\pi(\mathbf{x}))$, with a slight abuse of notation we have $\lim_{t\to\infty} p_t(\mathbf{x}_0,\mathbf{x}) \times \pi(\mathbf{x})^{-1} \to \pi(\mathbf{x})$. Interestingly, the expression $p_t(\mathbf{x}_0,\mathbf{x}) \times \pi(\mathbf{x})^{-1}$ can be interpreted as the transition density of a killed Brownian motion conditional upon its survival until time t (also called the quasi-stationary transition density) [Pollock et al., 2017]. Therefore, we can obtain samples from π by simulating a Brownian

motion which exhibits a killing mechanism of an instantaneous rate $\phi(\mathbf{x})$ at state $\mathbf{x}$. We elaborate more on this in the following subsection.

5.1.1 Quasi-Stationarity of a Killed Brownian Motion

Recall that the quasi-stationary density of a process is the limiting behaviour of its conditional density, given that the process survives. Formally, let $\{\mathbf{X}_t \,|\, t \geq 0\}$ with initial point $\mathbf{X}_0 = \mathbf{x}_0 \in \mathbb{R}^d$ denote a $d-$dimensional Brownian motion and a non-negative function $\kappa(\mathbf{x})$ (this will be defined momentarily in our context) denote the rate at which the process is killed at state $\mathbf{x}$. Further, assume that ζ is a random variable associated with potential killing time. If we define the conditional density given that our Brownian motion has survived until time t by

$$\mu_t(d\mathbf{x}) := \mathbb{P}(\mathbf{X}_t \in d\mathbf{x} \,|\, \zeta > t), \tag{5.1.5}$$

its quasi-stationary density is given by the limiting behaviour of μ_t as $t \to \infty$. In this setting, we define $\kappa(\cdot)$ under the following condition:

Condition–1 For $\beta(\mathbf{x}) := \nabla \log(\pi(\mathbf{x}))$, suppose there exists a finite $\Phi \in \mathbb{R}$ such that the hazard function defined by

$$\kappa(\mathbf{x}) := \phi(\mathbf{x}) - \Phi = \left(||\beta(\mathbf{X}_s)||^2 + \text{Div}\beta(\mathbf{X}_s)\right)/2 - \Phi \geq 0 \,\text{for all}\, \mathbf{x} \in \mathbb{R}^d. \tag{5.1.6}$$

As a result the transition density of a Brownian motion killed at rate $\kappa(\cdot)$ can be interpreted in the following sense:

Theorem 5.1.1 *Consider a standard Brownian motion $\{\mathbf{X}_t : t \geq 0\}$ which is killed with a state-dependent killing rate κ. Then the transition density of this killed Brownian motion conditional on its survival until time t is given by*

$$\mu_t(d\mathbf{x}_t) \propto \exp\left(-\frac{||\mathbf{x}_t - \mathbf{x}_0||^2}{2t}\right) \mathbb{E}_{\mathbb{W}}\left(\exp\left\{-\int_0^t \kappa(\mathbf{X}_s)ds\right\} \middle| \mathbf{X}_0 = \mathbf{x}_0, \mathbf{X}_t = \mathbf{x}_t\right) \tag{5.1.7}$$

Proof Refer to Appendix (A).

Pollock et al. [2017] shows that under certain regularity conditions, μ_t converges pointwise to π under the L^1 norm. Therefore, as motivated in Section (5.1), the problem of obtaining samples from the target distribution π translates to a good

choice of κ. We present the result below and refer to [Pollock et al., 2017] for conditions for convergence and a detailed proof.

Theorem 5.1.2 ([Pollock et al., 2017]) *Suppose there exist a constant* Φ *such that the hazard rate* $\kappa(\mathbf{x}) := \phi(\mathbf{x}) - \Phi$ *is always non-negative then* μ_t *converges to the target density* π *pointwise and under the* L^1 *norm.*

A recent advancement in the theory of QSMC by Wang et al. [2017] has relaxed the conditions for the convergence of Theorem (5.1.2) for a killed Brownian motion. Moreover, Wang et al. [2017] extends Theorem (5.1.2) to more general diffusion processes (see [Wang et al., 2017, Assumption 1] for details) and argues that a suitable $\kappa(\cdot)$ can be constructed so that the quasi-stationary distribution of the resulting killed diffusion coincides with π. We refer to [Wang et al., 2017] for the regularity conditions. However, [Wang et al., 2017] requires that the state space of the underlying killed Markov process is compact and that the corresponding killing rate is bounded. Recall that the algorithm (4.1.1) simulates the time of regeneration by uniformly sampling between the initial time and the time of kill. However, [Wang et al., 2017] extends this idea by simulating the time of regeneration from a suitably chosen beta distribution which leads to an 'optimal' convergence (see [Wang et al., 2017, Remark 2.2] for details). Intuitively, we should note that a choice of beta distribution forces the regeneration from recent history rather than the entire history of the chain.

Theorem (5.1.2) leads to the notion of a quasi-stationary Monte Carlo (QSMC) in which we consider a Brownian motion with appropriately chosen κ, whose quasi-stationary density is our target. Theorem (5.1.2) forms the basis of a QSMC method which provides an alternative mechanism to obtain draws from our target density π. As a result, one could simulate according to μ_t for sufficiently long time t and approximate π with μ_t. Hence, we need to be able to make exact draws from μ_t.

One method of sampling from μ_t is to simulate independent trajectories of a killed Brownian motion, which survives until time t and use its terminal position at time t as the realisation from μ_t. However, it should be noted from expression (5.1.7) that the probability of survival decreases exponentially with time t. Therefore, this approach of obtaining samples is highly infeasible. In the next section we discuss a method which can be implemented to obtain exact draws according to μ_t.

5.2 Constructing the QSMC draws

The exact simulation of the trajectories of a killed Brownian motion can be addressed using a path-space rejection sampler (see Section (3.2) for details). In this setting, we can propagate according to the transition density (5.1.7) by proposing a sample trajectory $\mathbf{X} = \{\mathbf{X}_s : 0 \leq s \leq t\}$ according to a Brownian motion and accepting it with probability

$$P(\mathbf{X}) = \exp\left\{-\int_0^t \kappa(\mathbf{X}_s)ds\right\}. \tag{5.2.1}$$

However, the simulation of a $P(\mathbf{X})$–coin is impossible since it requires us to have realised the entire sample path in the time interval $[0, t]$. We can circumvent this issue using a Poisson thinning approach, which requires us to only realise the sample path at a finite number of time points upto time t.

Let us assume that we use a Poisson process $\xi_1, \xi_2, ...$ to describe the potential killing times of a Brownian motion $\{\mathbf{X}_t : t \geq 0\}$. If $\kappa(\mathbf{x})$ denotes the instantaneous rate at which our process is killed at $\mathbf{x}$, then $\exp\left\{-\int_0^t \kappa(\mathbf{X}_s)ds\right\}$ represents the probability that the process survives along the trajectory $\{\mathbf{X}_t : t \geq 0\}$ until time t. However, it is impossible to simulate the event times of a non-homogeneous Poisson process directly. Therefore a Poisson thinning approach (outlined in algorithm (2.3.2)) can be used, which involves simulating a dominating Poisson process of rate K such that $\kappa(\cdot) \leq K$ and then independently simulating the positions of a Brownian motion at the realised events of our Poisson process. At each potential kill times ξ of the dominating Poisson process we simulate if the death occurs. The probability of death of a Brownian motion at a realised event time of ξ is equal to $\kappa(\mathbf{x}_\xi)/K$. The following result summarises the idea of a Poisson thinning:

Theorem 5.2.1 ([Pollock et al., 2017]) *Let $\xi_1, \xi_2, ...,$ be the realizations from a homogeneous Poisson process of rate $K \geq \sup_{\mathbf{X}_t | t \geq 0} \kappa(\mathbf{X}_t)$. Let $\mathbf{X}_{\xi_1}, \mathbf{X}_{\xi_2}, ...$ be the realizations of a Brownian motion $\mathbf{X} = \{\mathbf{X}_t : t \geq 0\}$ at $\xi_1, \xi_2, ...,$. If a Brownian motion is killed at ξ_i with probability $\frac{\kappa(X_{\xi_i})}{K}$ then,*

$$\mathbb{P}(\textit{BM survives until time } t \textit{ along the trajectory } \mathbf{X}) = \exp\left\{-\int_0^t \kappa(\mathbf{X}_s)ds\right\}. \tag{5.2.2}$$

5.2.1 An Importance Sampler for Drawing the Terminal Positions of a Killed Brownian Motion

In order to obtain the position of a killed Brownian motion, one can use Theorem (5.2.1) together with a rejection sampler. In this setting, we first simulate the potential killing times, say $\xi_1, \xi_2, ..., \xi_k$ within the interval $[0, t]$. If our Brownian motion is killed at any one of the potential death events this draw is rejected. Once Brownian motion survives all death events, the terminal position $\mathbf{x}_t$ is returned as a sample from π.

A rejection sampler as described above can often become inefficient for large values of t. This is because the acceptance probability can become negligible for large values of t. To circumvent this, we can use its importance sampling variant, where we attach the survival probability until time t as an importance weight to the terminal position $\mathbf{x}_t$. More formally, the terminal position $\mathbf{x}_t$ is assigned a weight of

$$W_t := \prod_{i=1}^{k} \left(1 - \frac{\kappa(\mathbf{x}_{\xi_i})}{K}\right). \tag{5.2.3}$$

In the presence of a positive lower bound of $\kappa(\cdot)$, say $K^{\downarrow}$, one can view a Brownian motion killed at rate $\kappa(\cdot)$ as a combination of two independent death processes of rate $K^{\downarrow}$ and $\kappa(\cdot) - K^{\downarrow}$ respectively [Pollock et al., 2017]. The survival of a Brownian motion until time t is guaranteed only if it survives both death processes. The survival probability of a Brownian motion when the hazard occurs at a constant rate $K^{\downarrow}$ is $\exp\{-K^{\downarrow}t\}$. In the latter case, a Brownian motion is killed at instance ξ_i (event times of a Poisson process of rate $\kappa(\cdot) - K^{\downarrow}$) with probability $\frac{\kappa(\mathbf{x}_{\xi_i}) - K^{\downarrow}}{K - K^{\downarrow}}$. Therefore, we can attach an importance weight of

$$W_t := \exp\{-K^{\downarrow}t\} \prod_{i=1}^{k} \left(1 - \frac{\kappa(\mathbf{x}_{\xi_i}) - K^{\downarrow}}{K - K^{\downarrow}}\right) \tag{5.2.4}$$

to our terminal position $\mathbf{x}_t$. However, we should note that the above approach requires us to find bounds K and $K^{\downarrow}$, which might not be possible in a more general setting. In this case we use the localised approach outlined in Section (3.3) in our context, which is elaborated in the following section.

5.2.2 Simulating Terminal Position of a Killed Brownian Motion using Localisation

Localisation allows us to weaken the condition of finding a global upper bound of the killing rate. Instead of a global bound, we focus over a compact interval. Under this approach, the sample path of a Brownian motion is simulated by first fixing a hypercube $\mathcal{H}_1$ around the initial position $\mathbf{x}_0$, which is followed by the simulation of the first passage time $\tau_1 = \{t > 0 : \mathbf{X}_t \notin \mathcal{H}_1\}$. This allows us to find the upper bound,$U^1_{\mathbf{x}} := \sup_{\mathbf{X}_t \in \mathcal{H}_1} \phi(\mathbf{X}_t)$ and the lower bound $L^1_{\mathbf{x}} := \inf_{\mathbf{X}_t \in \mathcal{H}_1} \phi(\mathbf{X}_t)$ of $\phi(\cdot)$, irrespective of whether the hazard rate is bounded globally or not (and subsequently $\kappa(\cdot)$), for all points inscribed within the hypercube $\mathcal{H}_1$. We can therefore find the rate of a dominating Poisson process within this hypercube and finally perform Poisson thinning within this interval. To achieve this, we can simulate a sequence of potential killing times $\xi_1, ..., \xi_k$ within the interval $[0, \min\{t, \tau_1\}]$, by means of a Poisson process of rate $U^1_{\mathbf{x}} - L^1_{\mathbf{x}}$. This follows due to the argument laid down in the earlier Section (5.2.1) and observing that $K = K_{\mathbf{x}} := U^1_{\mathbf{x}} - \Phi$ and $K^{\downarrow} = K^{\downarrow}_{\mathbf{x}} := L^1_{\mathbf{x}} - \Phi$. We then simulate the position $\mathbf{x}_\xi$ of a Brownian motion at the event time ξ, conditional on the layer information and the hypercube information (See Section (3.3) for details). Subsequently, a sequence of user-specified hypercubes $\{\mathcal{H}_2, \mathcal{H}_3, ..\}$ is fixed and followed by the simulation of a sequence of first passage times $\{\tau_2, \tau_3, ...\}$ in time order; this localises the path of a Brownian motion. We can then use an importance sampling algorithm to sample the terminal position of a killed Brownian motion by attaching an importance weight of

$$W_{\mathcal{H}_1} := \exp\left(-\left(L^1_{\mathbf{x}} - \Phi\right) \times \left(\min\{t, \tau_1\}\right)\right) \prod_{i=1}^{k} \frac{U^1_{\mathbf{x}} - \phi(\mathbf{x}_{\xi_i})}{U^1_{\mathbf{x}} - L^1_{\mathbf{x}}} \tag{5.2.5}$$

within the hypercube $\mathcal{H}_1$, where k is the number of potential kill times observed within the hypercube $\mathcal{H}_1$. Similarly, we can find importance weights $W_{\mathcal{H}_2}, W_{\mathcal{H}_3}, ...$ for the subsequent hypercubes, as given in equation (5.2.5). We present an importance sampler to simulate the terminal position of a killed Brownian motion together

with its importance sampling weight in Algorithm (5.2.1).

Algorithm 5.2.1: IMPORTANCE SAMPLING KILLED BROWNIAN MOTION$(\mathbf{X}_0, t)$

---

output ($\mathbf{X}_t, w_t^*$: Terminal position and weight of a killed Brownian motion)
set $i \leftarrow 1, j \leftarrow 0, \tau_0 \leftarrow 0, \xi_j \leftarrow 0$
while $\xi_j \leq t$
do
1. Fix hypercube $\mathcal{H}_i$ and calculate $L^i_{\mathbf{x}}, U^i_{\mathbf{x}}$
2. Simulate layer information $R^{(i)}_{\mathbf{x}} := (\tau_i, \mathbf{X}_{\tau_i})$
while $\xi_j < \tau_i$
do
3. Simulate $E \sim Exp(U^i_{\mathbf{x}} - L^i_{\mathbf{x}})$
4. Set $j = j + 1$ and $\xi_j = \min\{(\xi_{j-1} + E), \tau_i, t\}$
5. Set weight $w^*_{\xi_j} = w^*_{\xi_{j-1}} \times \exp(-(L^i_{\mathbf{x}} - \Phi)(\xi_j - \xi_{j-1})) \times \frac{(U^i_{\mathbf{x}} - \phi(\mathbf{X}_{\xi_j}))}{(U^i_{\mathbf{x}} - L^i_{\mathbf{x}})}$
6. Simulate positions $\mathbf{X}_{\xi_j} \sim MVN(\mathbf{X}_{\xi_{j-1}}, (\xi_j - \xi_{j-1})I_d) | R^{(i)}_{\mathbf{x}}$ (ref Section (3.3))
if $\xi_j = t$
then return $(\mathbf{X}_t, w_t^*)$
else if $\xi_j = \tau_i$
then $i \leftarrow i + 1$ and **go to** (1.)

We could repeat Algorithm (5.2.1) for N number of times and for a large value of t. The resulting weights can then be normalised and later used to approximate the quasi-stationary density of a killed Brownian motion. However, for a large value of t the variance of our importance weights can be large, which can be addressed using the SMC-based method outlined in Section (4.2.1).

To achieve this, we can employ Algorithm (5.2.1) as a proposal mechanism within our SMC framework (4.2.1). It discretises the time interval $[0, t]$ into m intervals for some user-specified values of t (the maximum diffusion time) and m (number of re-sampling time points) in order to construct the re-sampling points $t_i := it/m$ for $i = 1, ..., m$. We start the algorithm with N particles which are suitably initialised with weights equal to $1/N$. Then the position of a Brownian motion is realised for all N particles at the first re-sampling time t_1, using the importance sampling Algorithm (5.2.1); we denote the positions and its corresponding weights by $\mathbf{x}^{(1:N)}_{t_1}$ and $w^{*(1:N)}_{t_1}$ respectively. The importance weights are then normalised to produce weights within $(0, 1)$. If the effective sample size of N particles (say N_{eff}) is less than a given threshold (say N_{th} supplied by the user), particles are re-sampled

according to the empirical distribution of their weights. Upon re-sampling, newly sampled particles are assigned a weight of $1/N$. These set of N weighted particles are propagated until the re-sampling time t_2, together with the calculation of the importance weight at time t_2 as per Algorithm (5.2.1). This procedure is iterated until we have a set of weighted particles at our final time t.

The output using the above steps can then be used to estimate our target density π. A user-specified burn-in time t^* is usually fixed and samples falling within an equally specified grid of time interval $[t^*, t]$ are then collected to estimate the quasi-stationary density of our killed Brownian motion. The approximated density $\tilde{\pi}$ is given by the weighted occupation measure of the trajectories of N particles falling within the interval $[t^*, t]$. If we denote by m to be the number of re-sampling time points falling within the interval $[t^*, t]$ then the approximate density is given by

$$\tilde{\pi}(d\mathbf{x}) := \frac{1}{m} \sum_{t_i \in [t^*, t]}^{m} \sum_{j=1}^{N} w_{t_i}^{(j)} \times \delta_{\mathbf{X}_{t_i}^{(j)}}(d\mathbf{x}). \tag{5.2.6}$$

5.3 Using Sub-Sampling to Kill Brownian Motion: The ScaLE Algorithm

We observed in Section (5.2) that algorithm (5.2.1) can be used to simulate the position and importance weight of a killed Brownian motion at time t. This was dependent upon $\kappa(\mathbf{x})$ – the hazard rate of a killed Brownian motion. However, in a big-data setting where the posterior is of the form

$$\pi(\mathbf{x}) \propto f_0(\mathbf{x}) \prod_{i=1}^{n} f_i(\mathbf{x}), \tag{5.3.1}$$

the calculation of $\phi(\mathbf{x})$ at every potential death event is computationally infeasible. This is because $\phi(\mathbf{x})$ takes the following form:

$$\phi(\mathbf{x}) = \frac{\left|\left| \sum_{i=0}^{n} \nabla \log(f_i(\mathbf{x})) \right|\right|^2 + \sum_{i=0}^{n} \mathrm{Div} \nabla \log(f_i(\mathbf{x}))}{2}, \tag{5.3.2}$$

which requires us to sum the gradient and the divergence term over the full data set of size n, therefore, leading to a computational cost of $\mathcal{O}(n)$. To circumvent this problem, an unbiased estimator $\tilde{\phi}(\mathbf{x})$ of $\phi(\mathbf{x})$ can be constructed using sub-samples

of data which is computationally cheaper to evaluate. The unbiased estimator of $\phi(\cdot)$ can further be used to construct an unbiased estimator of the killing rate $\kappa(\cdot)$, while maintaining the exactness of our algorithm. For a formal treatment of the above ideas we introduce an auxiliary random variable $A \sim \mathcal{A}$ defined on the sample space $\{0, 1, ..., n\}$, which can be used to construct the unbiased estimator $\tilde{\phi}$ of ϕ, such that

$$\mathbb{E}\left(\tilde{\phi}_A(\cdot)\right) = \phi(\cdot). \tag{5.3.3}$$

Recalling that $\kappa(\cdot) = \phi(\cdot) - \Phi$, an unbiased estimator of the killing rate can be constructed as $\tilde{\kappa}_A(\cdot) := \tilde{\phi}_A(\cdot) - \Phi$. Here we should note that the non-negativity of $\tilde{\kappa}_A(\cdot)$ is ensured by a choice of a lower bound $\tilde{\Phi}$ of $\tilde{\kappa}_A(\cdot)$; this has been explained in Section (5.3.2).

Recall that, in Algorithm (5.2.1) we used Poisson thinning to simulate a Poisson process of rate $\kappa(\cdot)$. To achieve this, we first simulated the trajectory of a Brownian motion constrained within the hypercube $\mathcal{H}$. The hypercube construction was further used to bound the killing rate by finding a constant $K_{\mathbf{x}} \in \mathbb{R}_+$ such that $\kappa(\mathbf{x}) \leq K_{\mathbf{x}}$ for all $\mathbf{x} \in \mathcal{H}$. We then constructed our potential killing times by simulating a dominating Poisson process of rate $K_{\mathbf{x}}$ and killing the trajectory of a Brownian motion at $\mathbf{x}_\xi$ with probability $\kappa(\mathbf{x}_\xi)/K_{\mathbf{x}}$. The simulation of a Poisson process of rate $\kappa(\cdot)$ can also be achieved by constructing a dominating Poisson process of rate $\tilde{K}_{\mathbf{x}} \geq K_{\mathbf{x}}$. However, simulation of a dominating Poisson process of a higher rate $\tilde{K}_{\mathbf{x}}$ leads to an increase in the expected computational cost by a factor of $\tilde{K}_{\mathbf{x}}/K_{\mathbf{x}}$. This results in a larger number of potential killing events, where our Brownian motion has a smaller acceptance probability [Pollock et al., 2017].

Let us assume that within a given hypercube $\mathcal{H}$, we can construct an unbiased estimator $\tilde{\kappa}_A(\cdot)$ and evaluate the upper bound of $\tilde{K}_{\mathbf{x}} \in \mathbb{R}_+$ such that $0 \leq \tilde{\kappa}_A(\mathbf{x}) \leq \tilde{K}_{\mathbf{x}}$ for all $A \sim \mathcal{A}$ and $\mathbf{x} \in \mathcal{H}$. Then, an equivalent way of simulating the death events of rate $\kappa(\cdot)$ is to simulate a dominating Poisson process of rate $\tilde{K}_{\mathbf{x}}$ and determine the killing of the trajectory of a Brownian motion by realising $A \sim \mathcal{A}$ and killing the process at $\mathbf{x}_\xi$ with probability $\tilde{\kappa}_A(\mathbf{x}_\xi)/\tilde{K}_{\mathbf{x}}$. In such a setting, we should be able to find an auxiliary random variable $A \sim \mathcal{A}$ and an unbiased estimator which can be evaluated without looping over the entire data set. We also need to make sure that the ratio $\tilde{K}_{\mathbf{x}}/K_{\mathbf{x}}$ is not too large as this could result in the simulation of too many death events.

5.3.1 Constructing an Unbiased Estimator of the Killing Rate

We have observed that a decision on the kill/survival of a Brownian motion can be also carried out by constructing an unbiased estimator of the hazard rate. Therefore, we lay down the construction of an unbiased estimator, which is followed by the computation of its bounds in order to perform Poisson thinning within a given hypercube. Let us recall that $\log \pi(\mathbf{x}) = \sum_{i=0}^{n} \log f_i(\mathbf{x})$ and assume $\mathcal{A}$ to be the law of $I \sim U\{0, 1..., n\}$. We define

$$\alpha(\mathbf{x}) := \nabla \log \pi(\mathbf{x}) - \nabla \log \pi(\hat{\mathbf{x}}), \tag{5.3.4}$$

and its unbiased estimator as

$$\tilde{\alpha}_I(\mathbf{x}) := (n+1) \left(\nabla \log f_I(\mathbf{x}) - \nabla \log f_I(\hat{\mathbf{x}}) \right), \tag{5.3.5}$$

where $\hat{\mathbf{x}}$ often called *control variate*, which is a point closer to the mode of the posterior distribution π. Since the posterior distribution contracts at the rate of $\mathcal{O}(n^{-1/2})$, point $\hat{\mathbf{x}}$ needs to be within $\mathcal{O}(n^{-1/2})$ of the true posterior mode to achieve $\mathcal{O}(1)$ iterative computational cost [Pollock et al., 2017]. Following the above notions, we can construct an unbiased estimator of $\nabla \log \pi(\mathbf{x})$ as $\nabla \log \pi(\hat{\mathbf{x}}) + \tilde{\alpha}_I(\mathbf{x})$. The advantage of constructing such an unbiased estimator instead of $(n+1)\left(\nabla \log f_I(\mathbf{x})\right)$ can be validated by the following argument: in our case $\nabla \log \pi(\mathbf{x})$ is a smooth function of $\mathbf{x}$, therefore,

$$\nabla \log f_I(\mathbf{x}) \approx \nabla \log f_I(\hat{\mathbf{x}}), \text{ implying } \tilde{\alpha}_I(\mathbf{x}) \approx 0 \text{ whenever } \mathbf{x} \approx \hat{\mathbf{x}}. \tag{5.3.6}$$

Hence, $\nabla \log \pi(\hat{\mathbf{x}}) + \tilde{\alpha}_I(\mathbf{x})$ will have a smaller variance as compared to $(n+1)\left(\nabla \log f_I(\mathbf{x})\right)$, this is because $\nabla \log f_I(\mathbf{x})$ and $\nabla \log f_I(\hat{\mathbf{x}})$ will be correlated whenever $\mathbf{x} \approx \hat{\mathbf{x}}$. One good choice of $\hat{\mathbf{x}}$ is the posterior mode, however this is usually unknown, therefore, we require $\hat{\mathbf{x}}$ to be within $\mathcal{O}(n^{-1/2})$ distance of the posterior mode. Note that the calculation of $\hat{\mathbf{x}}$ can be done using a gradient descent method in order to find a value in the vicinity of the true posterior [Pollock et al., 2017]. Often its popular subclass called stochastic gradient decent method is used to calculate $\hat{\mathbf{x}}$ (see [Robbins and Monro, 1951; Bouleau and Lépingle, 1994; Duchi et al., 2011] for instance).

Using the expression for $\alpha(\cdot)$, the function $\phi(\cdot)$ can be re-expressed as

$$\phi(\mathbf{x}) = \left(\alpha(\mathbf{x})^T(\alpha(\mathbf{x}) + 2\nabla \log \pi(\hat{\mathbf{x}})) + \text{Div } \alpha(\mathbf{x})\right)/2 + C \text{ where,} \tag{5.3.7}$$

$$C := (||\nabla \log \pi(\hat{\mathbf{x}})||^2 + \text{Div} \log \pi(\hat{\mathbf{x}}))/2. \tag{5.3.8}$$

Observing that for independent $I, J \sim U\{0, ..., n\}$, an unbiased estimator of $\phi(\cdot)$ can be constructed as

$$\tilde{\phi}_A(\mathbf{x}) := \left(\tilde{\alpha}_I(\mathbf{x})^T(\tilde{\alpha}_J(\mathbf{x}) + 2\nabla \log \pi(\hat{\mathbf{x}})) + \text{Div } \tilde{\alpha}_I(\mathbf{x})\right)/2 + C. \tag{5.3.9}$$

Although the calculation of C requires $\mathcal{O}(n)$ computational complexity, it only needs to be done once, which can also be achieved completely in parallel [Pollock et al., 2017]. (see Karniadakis and Kirby II [2003]; Eddelbuettel [2013, 2018]; Matloff [2015] for the implementation of parallel architectures.) We can now emplace this newly constructed unbiased estimator within algorithm (5.2.1), which can be used within our SMC framework to produce the Scalable Langevin Exact (ScaLE) algorithm. Our final task is to construct the upper and the lower bound of the hazard rate $\phi(\cdot)$ which is used within Algorithm (5.2.1).

5.3.2 Computing Bounds of the Killing Rate

The implementation of the ScaLE algorithm requires us to bound the unbiased killing rate $\tilde{\kappa}_A(\cdot)$ within a given hypercube $\mathcal{H}$. We can achieve this by bounding the function $\tilde{\phi}_A(\cdot)$. Therefore, we demonstrate the construction of the upper and the lower bound of function $\phi_A(\cdot)$ within a given hypercube $\mathcal{H}$.

To compute this, we first consider the upper bound the second derivative of the log-likelihood function for each datum. More formally, we assume that the spectral radius of the second derivative follows

$$\rho(\nabla\nabla^T \log f_I(\mathbf{x})) \leq P_n, \tag{5.3.10}$$

for some $P_n > 0$. We use this assumption to bound $\tilde{\phi}_A(\cdot)$ both above and below. This can be achieved by upper bounding $\left|\tilde{\phi}_A(\cdot)\right|$ over all possible realisations of $A \sim \mathcal{A}$. Under the condition (5.3.10), the unbiased estimator $\tilde{\alpha}_I(\cdot)$ satisfies

$$\max_{\mathbf{x}\in\mathcal{H}, I\in\{0,...,n\}} |\tilde{\alpha}_I(\mathbf{x})| \leq (n+1) \cdot P_n \cdot \max_{\mathbf{x}\in\mathcal{H}} ||\mathbf{x} - \hat{\mathbf{x}}||. \tag{5.3.11}$$

We observe that for $C = (||\nabla \log \pi(\hat{\mathbf{x}})||^2 + \text{Div} \log \pi(\hat{\mathbf{x}}))/2$, the bound on the function

$$2\left|\tilde{\phi}_A(\mathbf{x}) - C\right| = \left(\tilde{\alpha}_I(\mathbf{x})^T(\tilde{\alpha}_J(\mathbf{x}) + 2\nabla \log \pi(\hat{\mathbf{x}})) + \text{Div } \tilde{\alpha}_I(\mathbf{x})\right), \tag{5.3.12}$$

which satisfies

$$2 \max_{\mathbf{x}\in\mathcal{H}, A\in\mathcal{A}} \left|\tilde{\phi}_A(\mathbf{x}) - C\right| \leq (n+1)\cdot P_n \cdot \max_{\mathbf{x}\in\mathcal{H}} ||\mathbf{x} - \hat{\mathbf{x}}|| \Bigg((n+1)\cdot P_n \cdot \max_{\mathbf{x}\in\mathcal{H}} ||\mathbf{x} - \hat{\mathbf{x}}|| + 2|\nabla \log \pi(\hat{\mathbf{x}})| \Bigg) + (n+1)\cdot P_n \cdot d \cdot \max_{\mathbf{x}\in\mathcal{H}} ||\mathbf{x} - \hat{\mathbf{x}}||. \tag{5.3.13}$$

Under the condition that the given posterior distribution contracts at a rate $n^{-1/2}$, the upper bound in the equation (5.3.13) is of $\mathcal{O}_p(1)$[1], which makes the intensity of the dominating Poisson process independent of n (ref to Appendix (D) for a detailed calculation). We refer to [Pollock et al., 2017, Proposition 2] for its formal discussion and the iterative computational complexity of the ScaLE algorithm.

5.4 A Note on the Choice of Parameters within the ScaLE Algorithm

Recall from the algorithmic construction of the ScaLE method that a user has the freedom to tune parameters such as N (number of particles; its discussion has been presented in Section (4.3)) and the hypercube sides $2\theta^{(i)}$. For a symmetric hypercube around a given point, its construction is equivalent to a choice of its side length $2\theta^{(i)}$ ($\theta^{(i)}$ is often called its *Layer size*) along the i^{th} dimension. However, it should be noticed that an arbitrary choice of side lengths can lead to computational bottleneck. We explain this using two situations a user can find themselves into.

A significantly small value of $\theta^{(i)}$ would force our Brownian motion to exit the hypercube more frequently, therefore, requiring frequent simulations of the exit times which can be computationally cumbersome. Therefore, a choice of very small values of $\theta^{(i)}$ should be avoided. Secondly, we observe from expression (5.3.13) that the upper bound of the killing rate $\kappa(\cdot)$ is an increasing function of the side lengths. Hence, a large choice of side lengths would result into a substantially increased number of potential death events, thus leading to an increase in the overall computational cost.

[1] A sequence of random variable X_n is $\mathcal{O}_p(1)$ if for every $\epsilon > 0$ there exist N and δ such that $P(|X_n| \geq \delta) \leq \epsilon$ for all $n \geq N$.

A good choice of the side lengths are therefore essential to minimizing the overall computational complexity of the ScaLE algorithm.

The major computational effort within a QSMC method can be attributed to the simulation of the potential killing times and the positions of a killed Brownian motion. This cost is a function of bounds of the hazard rate within a given hypercube. We can observe from expression (5.3.13) that this bound increases at least linearly as a function of the number of dimensions d. Therefore, the computational cost of a QSMC method is at least of $\mathcal{O}(d)$ as a function of the number of dimensions. However, we should note that this computational burden can be further reduced by allowing sub-sampling among dimensions (see [Pollock et al., 2017]).

Part II

Methodology and Applications

Chapter 6

The Regenerating Scalable Langevin Exact Method

In Chapter (5) we reviewed a methodology developed in [Pollock et al., 2017] which can be used to sample exactly according to an intractable posterior density in the context of a Bayesian inference for tall data. Contrary to the traditional MCMC methods, the resulting Quasi-Stationary Monte Carlo (QSMC) method is based on the quasi-stationary distribution of an appropriately constructed stochastic process. Later, we realised that the quasi-stationary density of a Brownian motion could be chosen as a proxy for the underlying intractable density. This quasi-stationary density was approximated using a sequential importance re-sampling method. In this context, a large number of weighted particles were propagated over time, while attaching importance weights to each particle. The re-sampling step was then invoked to replicate particles with large importance weights. These combinations of sampling and re-sampling were carried out for a sufficiently long time in order to guarantee the convergence to the quasi-stationary density of a killed Brownian motion. The resulting Scalable Langevin Exact (ScaLE) algorithm produces an impressive sub-linear cost with respect to the data size when the true hazard rate is replaced by its unbiased estimator. However, we should note that the ScaLE method relies upon a large number of particles to start with, which can be computationally challenging to handle. At the same time, for a large number of particles, the re-sampling step can further gravitate the computational burden. These problems make the ScaLE algorithm computationally expensive. Therefore, we try and address these problems by considering an alternative mechanism to simulate according to the quasi-stationary density of a killed Brownian motion.

As mentioned above, this chapter takes a different approach to target the quasi-stationary density of a killed Brownian motion. Here we use a natural extension of the regenerating approach developed in [Blanchet et al., 2016]. As outlined in Chapter (4), this methodology approximates the quasi-stationary density of a killed Markov process by first propagating according to its dynamics until the process is killed. Secondly, upon killing, it regenerates itself at the instance of a kill by drawing uniformly along the path visited by the process. The combinations of the steps above are repeated until a sufficiently long time t for it to converge to the quasi-stationary density. Therefore, we use this regenerating approach in our case i.e. for a Brownian motion, thereby naming our resulting method as the *Regenerating ScaLE (ReScaLE)* method. We empirically validate that the ReScaLE method can be applied in a Bayesian inference for a tall data problem. Similar to ScaLE, the ReScaLE method achieves the sub-linear scalability with respect to the data size. However, we should note that we do not compare the algorithmic performance of these QSMC methods in this thesis.

This chapter is organised as follows: Section (6.1) first revisits the theory of QSMC briefly, which is followed by the construction of the ReScaLE method in a situation where the hazard rate is bounded. We further consider two applications of the ReScaLE algorithm: one to a synthetic data while another concerns a real dataset in Section (6.2). Section (6.3) outlines the construction of the ReScaLE algorithm for an unbounded hazard rate. We discuss the scalability of the ReScaLE method in Section (6.4). The following Section (6.5) considers an application to the US domestic airline dataset to illustrate the working of the ReScaLE methodology in the Bayesian inference on a big dataset. In subsection (6.5.2) we illustrate the working of the ReScaLE method on a synthetic data where the posterior assumes a non-normal density. This is followed by an empirical illustration of the scalability of the ReScaLE method with respect to the data size.

6.1 Constructing Quasi-Stationary Monte Carlo Draws

Recall that, Theorem (5.1.2) guarantees that for an intractable distribution $\pi(\cdot)$, a Brownian motion $\{\mathbf{X}_t : t \geq 0\}$ which is killed at state $\mathbf{x}$ at the rate $\kappa(\mathbf{x}) = \phi(\mathbf{x}) - \Phi \geq 0$, then the quasi-stationary density

$$\lim_{t\to\infty} \mu_t(d\mathbf{x}) = \lim_{t\to\infty} \mathbb{P}(\mathbf{X}_t \in d\mathbf{x} \,|\, \zeta > t) \tag{6.1.1}$$

can be used as a proxy for the target density $\pi(\mathbf{x})$. This in turn gives us a mechanism to sample values according to π, by simulating according to the quasi-stationary density of a killed Brownian motion. Therefore, in order to simulate according to this quasi-stationary density, we need to be able to simulate the trajectories of a killed Brownian motion according to its transition density

$$\mu_t(\mathbf{x}_t) \propto \exp\left(-\frac{||\mathbf{x}_t - \mathbf{x}_0||^2}{2t}\right) \mathbb{E}_{\mathbb{W}}\left(\exp\left\{-\int_0^t \kappa(\mathbf{X}_s)ds\right\}\middle| \mathbf{X}_0 = \mathbf{x}_0, \mathbf{X}_t = \mathbf{x}_t\right), \tag{6.1.2}$$

for a sufficiently long time t. Hence, this section is devoted to the construction of a sample path according to the dynamics of a killed Brownian motion (6.1.2). As mentioned before, we employ a natural extension of the regenerating approach to a general state-space setting in order to sample from the quasi-stationary density of a killed Brownian motion (see Chapter (4)). Here, we consider the two cases separately; one when the hazard rate is bounded and second when it is unbounded.

6.1.1 When Hazard Rate is Bounded

We first consider the case of a bounded hazard rate; we assume that there exist a positive constant K such that $\kappa(\mathbf{x}) \leq K$ for all $\mathbf{x} \in \mathbb{R}^d$. A method for simulating such a trajectory which is killed at a hazard rate $\kappa(\cdot)$, is presented in the paragraph below.

In order to simulate the sample trajectory of a killed Brownian motion, Theorem (5.2.1) suggests that we simulate all potential death events up to time t. We then simulate the position of a Brownian motion in time-order and decide whether or not the process dies. We continue this until either the process dies or survives all potential death events in the interval $[0, t]$. Our Brownian motion must be able to survive for a sufficiently long time in order to obtain samples from the target density. However, the survival of a Brownian motion is highly unlikely as the survival probability decreases exponentially with time. To circumvent this issue, we employ the regenerative mechanism outlined in the chapter (4). The simplicity of this approach makes the sampling from the quasi-stationary density feasible. In the following subsection, we combine ideas outlined so far with respect to the ReScaLE algorithm for the case where the hazard rate is bounded.

6.1.2 ReScaLE Algorithm - Pseudocode for Bounded Hazard Rate

Theorem (5.2.1) helps us to simulate the potential death events $\{\xi_i\}_{i\geq 1}$ for our path, using the auxiliary Poisson process of rate K. We simply have to identify the first accepted event; this is our *killing time* t_{kill}. More formally,

$$t_{kill} := \min\{\xi_j \,:\, \xi_j > 0, U_j \leq \kappa(\mathbf{x}_{\xi_j})/K, U_j \sim U[0,1]\}. \tag{6.1.3}$$

Once our Brownian motion is killed at some point t_{kill}, we can regenerate its path by simulating from the occupation measure. Recall, the occupation measure is simply

$$\mu_{t_{kill}}(B) := \frac{1}{t_{kill}} \int_0^{t_{kill}} \mathbb{I}_B(\mathbf{X}_s) ds, \quad \text{for } B \subset \mathbb{R}^d. \tag{6.1.4}$$

We can simulate according to the occupation measure by further noting that for $U \sim U[0, t_{kill}]$,

$$\mathbb{E}_U \left(\mathbb{I}_B(\mathbf{X}_s) \,|\, \mathbf{X}\right) = \mu_{t_{kill}}(B). \tag{6.1.5}$$

If we denote the position of regeneration by $\mathbf{X}_{reg} \in \mathbb{R}^d$, the simulation of $\mathbf{X}_{reg}$ from the occupation measure can be performed by first simulating the time of regeneration $t_{reg} \sim U[0, t_{kill}]$. This is followed by the simulation of the position of our Brownian motion at t_{reg}, conditional on all information between 0 and t_{kill}. If $\xi_{j-1} < t_{reg} < \xi_j$ for some j, then the Markov property allows us to simulate $\mathbf{X}_{reg}$ simply using the law of a Brownian bridge between $(\xi_{j-1}, \mathbf{x}_{\xi_{j-1}})$ and $(\xi_j, \mathbf{x}_{\xi_j})$. This idea has been summarized in Algorithm (6.1.1).

Algorithm 6.1.1: ReScaLE Algorithm – Bounded Hazard($\mathbf{X}_0, t, \mathbf{m}$)

input t : maximum time, $\mathbf{X}_0$: initial position, a time mesh $\mathbf{m} : m_1 < ... < m_n < t$
global κ, K
output ($\mathbf{X}_{m_1}, ..., \mathbf{X}_{m_n}$: positions of the killed Brownian motion.)
set $i \leftarrow 0; \xi_i \leftarrow 0; \mathbf{X}_{\xi_i} \leftarrow \mathbf{X}_0; \mathbf{S} \leftarrow \emptyset$
while $\xi_i < t$

do
- KILL $\leftarrow$ **false**
- **while** KILL $=$ **false**
  - **do**
    - Simulate $E \sim Exp(K)$ and set $i \leftarrow i + 1$
    - $\xi_i = \xi_{i-1} + E$
    - $\mathbf{X}_{\xi_i} \sim MVN(\mathbf{X}_{\xi_{i-1}}, EI_d)$
    - $\mathbf{S} \leftarrow \mathbf{S} \cup (\xi, \mathbf{X}_{\xi_i})$
    - Simulate $U \sim U[0, 1]$
    - **if** $U \leq \kappa(\mathbf{X}_{\xi_i})/K$
      - **then**
        - KILL $\leftarrow$ **true**
        - $t_{kill} \leftarrow \xi_i$
- Simulate $t_{reg} \sim U[0, t_{kill}]$ and find j such that $\xi_{j-1} < t_{reg} < \xi_j$
- $\mathbf{x}_{reg} \sim MVN\left(\mathbf{X}_{\xi_{j-1}} + \frac{(t_{reg}-\xi_{j-1})}{(\xi_j-\xi_{j-1})}(\mathbf{X}_{\xi_j} - \mathbf{X}_{\xi_{j-1}}), \frac{(\xi_j-t_{reg})(t_{reg}-\xi_{j-1})}{(\xi_j-\xi_{j-1})} I_d\right)$
- $\mathbf{S} \leftarrow \mathbf{S} \cup (t_{reg}, \mathbf{x}_{reg})$ and set $\mathbf{X}_{\xi_i} \leftarrow \mathbf{x}_{reg}$

Evaluate Brownian bridge position at times $m_1, ..., m_n$ given $\mathbf{S}$
return ($\mathbf{X}_{m_1}, ..., \mathbf{X}_{m_n}$)

Figure (6.1) depicts the steps of Algorithm (6.1.1) in the univariate case. The final step of our Algorithm (6.1.1) finds the position of a Brownian motion at an independent time mesh $m_1, ..., m_n$ which can be simulated using the law of a Brownian bridges process. Figure (6.2) illustrates one such instance.

The output from Algorithm (6.1.1) can be then used to estimate the target density π by ignoring for the burn-in. We can specify the burn-in time t^* and the samples falling within an equally specified grid of time interval $[t^*, t]$ can then be collected to estimate our target density. The approximated density $\tilde{\pi}$ is given by the occupation measure of the trajectories of m^* sample points falling within the interval $[t^*, t]$. We

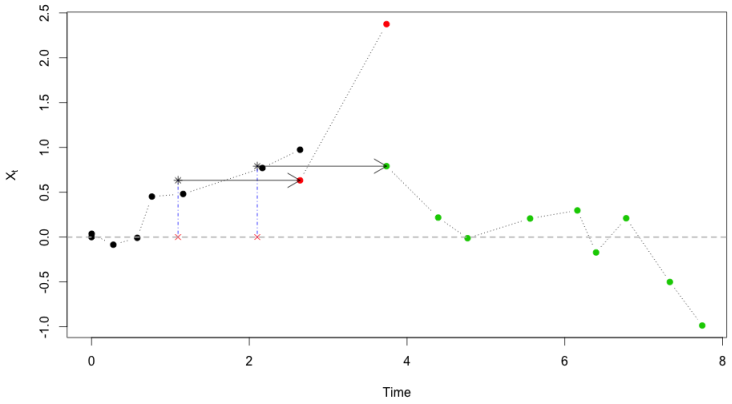


Figure 6.1: *This illustrates segments simulated using the ReScaLE algorithm for a bounded $\kappa(\cdot)$. It captures the situation when the process is killed three times. Here, we initialize the algorithm at the origin and continue sampling from a Brownian motion at the event times of a Poisson process of rate K until it gets killed. Killing in the first segment has been depicted through the termination of the black chain. Once the Brownian motion is killed, it regenerates by first simulating uniformly between 0 and t_{kill} and then simulating the Brownian motion at the observed time point. The regenerating time is denoted by a red cross that corresponds to the x-axis value of the rear part of the arrow connected to the first red dot. Once the starting time is observed it simulates the value of Brownian motion by simulating from a Brownian Bridge, conditioned on the two nearest neighbours of the observed regenerating time. It has been denoted by the first black star in the figure. This point now becomes the starting position for the next segment and the process continues in a similar manner.*

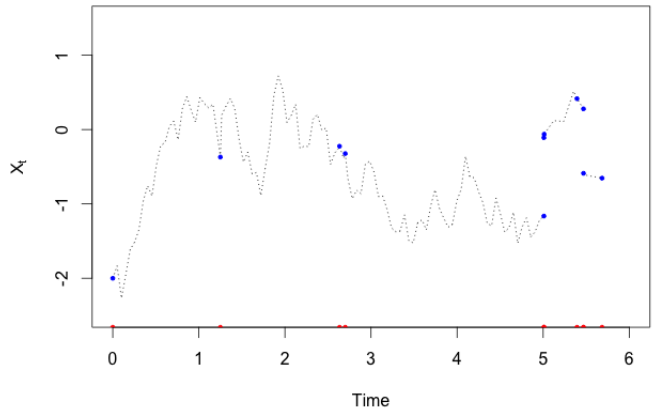


Figure 6.2: *Figure illustrates positions of a Brownian motion simulated at an equally-distant time mesh. Red points on time-axis are observed using a Poisson process of rate K while blue points denote the corresponding positions.*

approximate our target density using

$$\tilde{\pi}(d\mathbf{x}) := \frac{1}{m^*} \sum_{m_i \in [t^*, t]}^{m^*} \delta_{\mathbf{X}_{m_i}}(d\mathbf{x}). \tag{6.1.6}$$

6.2 Experimental Studies I – Bounded Hazard Rate

In this section we consider two examples where our aim is to sample from the posterior distribution of some parameters of interest. Our first example considers 5 data points simulated according to a Cauchy distribution, where we are interested in sampling from the posterior distribution of its location parameter. Our second example considered here is based on a real data set obtained from the `MASS` package in `R` [Venables and Ripley, 1994]. In this case we aim to sample from the posterior distribution of parameters of a logistic regression model.

6.2.1 Applying the ReScaLE Algorithm - a Toy Example

We consider a toy example consisting of 5 data points y_i which are simulated from the density

$$\pi(y|x) \propto \frac{1}{1 + (y - x)^2}, \tag{6.2.1}$$

where the true state of parameter x is 1. The simulated data is given as the following:

```
> y
[1] 2.65226687 1.27648783 1.61011759 1.27433040 0.08721209.
```

We choose a standard Cauchy prior for the location parameter x. Here, we are interested in simulating from the posterior density of the parameter x, which is given by

$$\pi(x|\mathbf{y} = (y_1, ..., y_5)) \propto \frac{1}{1 + x^2} \prod_{i=1}^{5} \frac{1}{1 + (y_i - x)^2}. \tag{6.2.2}$$

The hazard rate κ is depicted in Figure (6.3). It is also clear from Figure (6.3) that $K := \max_{x \in \mathbb{R}} \kappa(x) \leq 14$ (ref to Appendix (B.1) for a detailed numerical approximation). Here the ReScaLE algorithm was allowed to run for $t = 10^5$ after initialising at

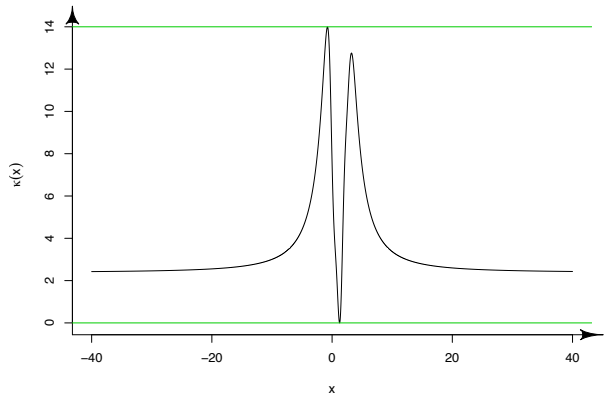


Figure 6.3: *ReScaLE for Cauchy simulated data: This illustrates the 'rate of kill' function $\kappa(x)$ together with its global extrema.*

$x_0 = 0$. A typical short and long run of the ReScaLE method is presented in the Figure (6.4) and Figure (6.5) respectively.

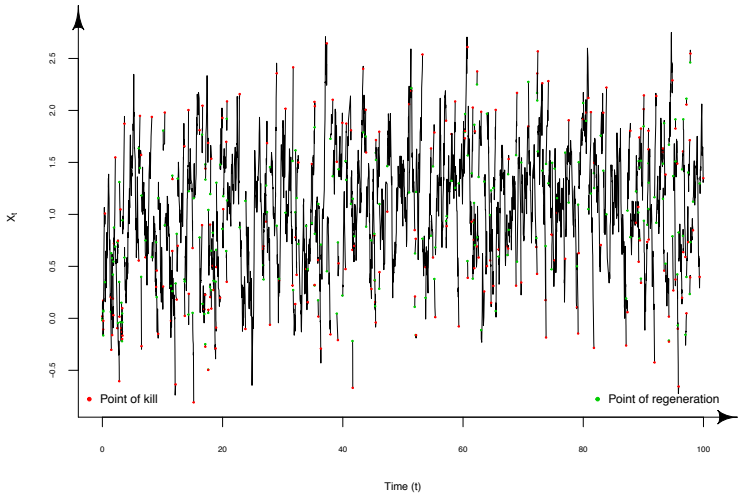


Figure 6.4: *ReScaLE for Cauchy simulated data: This illustrates a typical short run of the ReScaLE method for the toy problem considered. The points in red and green denote the position of kill and regeneration respectively.*

Figure (6.6) is a comparison of the kernel density obtained using the ReScaLE algorithm, to that of a numerically approximated density π. We numerically approximate the normalising constant (ref to Appendix (B.2) for a detailed description) of

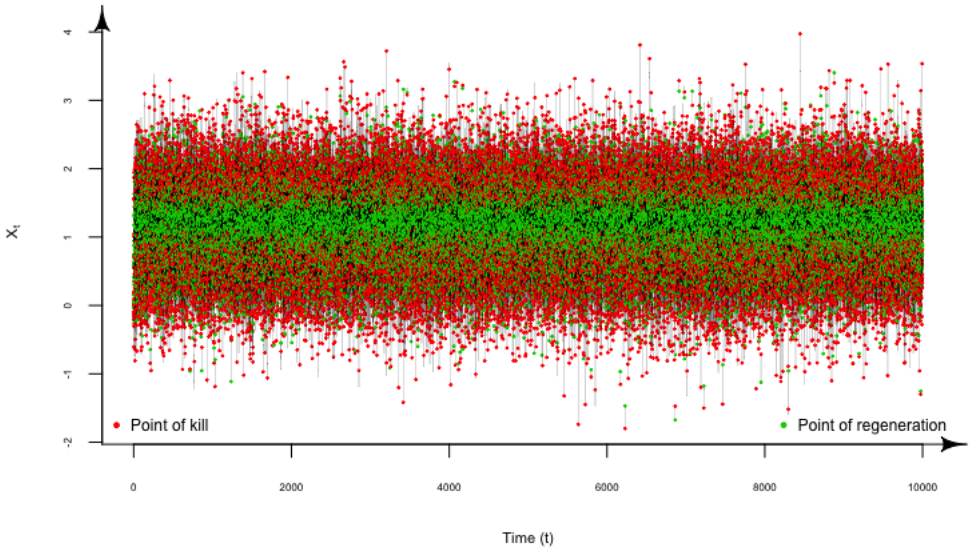


Figure 6.5: *ReScaLE for Cauchy simulated data: This figure illustrates a run of the ReScaLE method for the toy problem considered in this section. The algorithm was initialised at* $X_0 = 0$.

the posterior density which is given by

$$\pi(x|\mathbf{y} = (y_1, ..., y_5)) = \frac{1}{0.06281079} \frac{1}{1+x^2} \prod_{i=1}^{5} \frac{1}{1+(y_i - x)^2}. \tag{6.2.3}$$

We plot the kernel density approximation together with the numerically approximated density (6.2.3) in Figure (6.6). It can be observed visually that the ReScaLE algorithm approximates the true density really well.

Diagnosing the Quality of the Output

Here we examine how well the chain obtained using the ReScaLE method explores the support of the posterior distribution of interest. We also test if the kernel density approximated using the ReScaLE method is 'close' to the numerically approximated posterior density. We first estimate the integrated autocorrelation time in order to obtain a set of independent samples from the chain; this is estimated at 5.031489. Informally, this means that every 5th sample in the chain will be roughly independent of each other. For simplicity, we collect $\{X_j : j = 1 + 10i, j < N\}$ to diagnose the convergence of the empirical distribution $F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{I}_{(-\infty,x]}(X_i)$ to

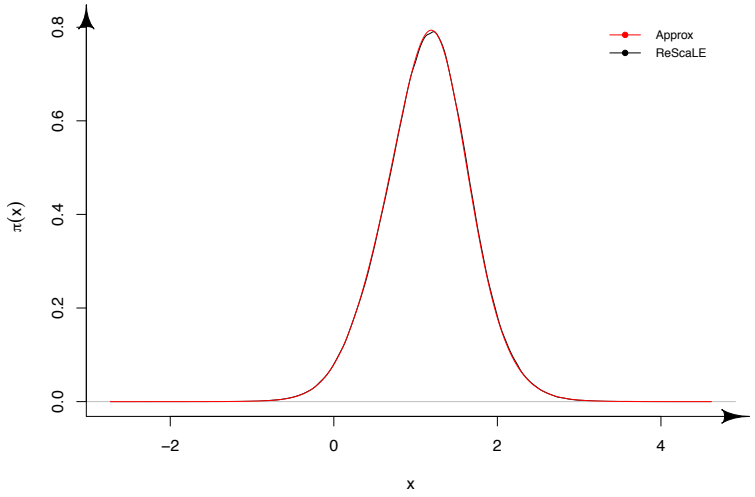


Figure 6.6: *ReScaLE for Cauchy simulated data: This illustrates the kernel density approximation of samples obtained using the ReScaLE algorithm. Red curve denotes the numerically approximated posterior density* $\pi(x|\mathbf{y})$.

the numerically approximated posterior distribution $F(x)$. We use the Kolmogorov-Smirnov test statistic as the measure of discrepancy between the empirical distribution and the numerically approximated posterior distribution. In this situation, the Kolmogorov-Smirnov test checks if the samples $\{X_j : j = 1 + 10i, j < N\}$ are drawn from the true posterior distribution. The p-value corresponding to this hypothesis testing is 0.491, which implies that the null hypothesis is not rejected at a 5% level of significance. Therefore, there is enough evidence to accept the fact that the kernel density approximation based on samples obtained via the ReScaLE method approximates the true posterior distribution very well. We present the uniform norm distance between the empirical distribution and approximate distribution using (6.2.3) in Table (6.1). Table (6.1) further illustrates the efficiency of 10 runs of the ReScaLE algorithm. Furthermore, we use `coda::gelman.diag()` function to estimate the potential scale reduction factor, which is 1.000007 in this example [Plummer et al., 2006]. It suggests that our ReScaLE chain has fully explored the support of the posterior density. We present the evolution of the potential scale reduction factor (PSRF) as a function of the diffusion time in Figure (6.7), which is obtained using the `R` function `coda::gelman.plot()`. The PSRF settles to a value close to 1 after a diffusion time of $\approx 2 \times 10^3$, which signifies the convergence to our target density.

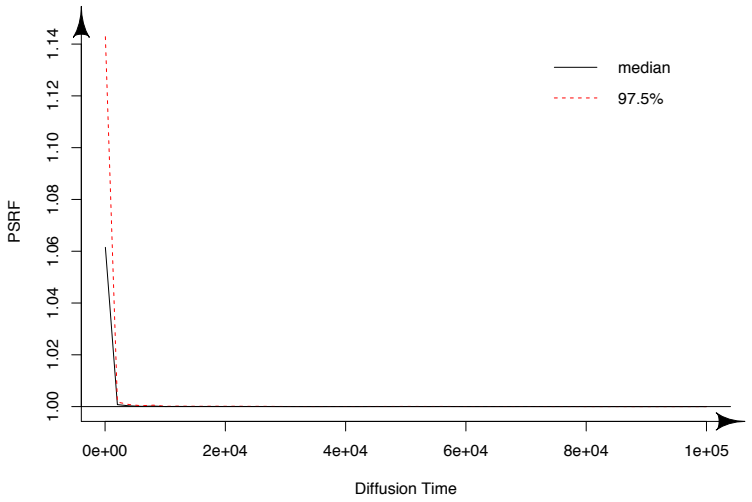


Figure 6.7: *ReScaLE for Cauchy simulated data: This figure illustrates the evolution of the potential scale reduction factor (2.5.9) with varying values of diffusion time t based on 10 runs of the ReScaLE algorithm. The solid line represents the median value of PSRF while the dotted line is the upper 97.5% confidence interval. The absence of fluctuation in this graph after $t \approx 2 \times 10^3$ shows that our method has converged to the target density.*

6.2.2 Logistic Regression : Age of Menarche in Warsaw

Next, we consider a 2-dimensional posterior distribution for a real world dataset. The `menarche` dataset under consideration is part of the `MASS` package in `R` [Venables and Ripley, 1994]. This dataset consists of a proportion of female adolescence at various age groups who reach menarche. This dataset has been previously used by [Aranda-Ordaz, 1981] and [Venables and Ripley, 1994]. The dependent variable y_i describes whether or not (1/0) the i^{th} individual reaches menarche, for a total of $n = 3918$ individuals. We consider the normalized `Age` as a continuous covariate and fit a logistic regression model of the form,

$$y_i = \begin{cases} 1 & \text{with probability} \quad p_i := 1/(1+\exp\left(-\beta_0 - \beta_1 \texttt{Age}_i\right)), \\ 0 & \text{otherwise.} \end{cases} \tag{6.2.4}$$

Letting $\beta := (\beta_0, \beta_1)$, our aim is to obtain samples from

$$\pi(\beta) \propto \prod_{i=1}^{n} p_i(\beta)^{y_i} \cdot (1 - p_i(\beta))^{1-y_i}, \tag{6.2.5}$$

Run	UND	Efficiency
1	0.0031	729.0138
2	0.0026	857.8111
3	0.0028	765.0388
4	0.0020	685.2588
5	0.0024	629.0287
6	0.0029	802.7332
7	0.0012	674.0973
8	0.0019	754.1556
9	0.0025	711.5874
10	0.0027	797.2982
Average	0.0024	740.6023

Table 6.1: *ReScaLE for Cauchy simulated data: This table presents the estimated uniform norm distance (UND, (2.5.2)) (between the empirical distribution function resulting due to the ReScaLE method and the numerically approximated distribution function using (6.2.3)) and the efficiency (ESS/sec) (2.5.5) for 10 different runs of the ReScaLE algorithm on the toy problem considered.*

for which the corresponding $\phi(\cdot)-$ function takes the form (ref to Appendix (C) for detailed calculations),

$$\phi(\beta) = \frac{1}{2}\left(\left(\sum_{i=1}^{n}(y_i - p_i)\right)^2 + \left(\sum_{i=1}^{n}(y_i - p_i)\cdot \texttt{Age}_i\right)^2 - \sum_{i=1}^{n} p_i(1-p_i)(1+\texttt{Age}_i^2)\right), \tag{6.2.6}$$

which is bounded in each component of β. However, we should note that the bound on $\phi(\cdot)$ can be extremely large in a tall data setting; we consider one such instance in Section (6.5).

We use the `glm()` function in `R` to estimate $\hat{\beta} = (1.410426, 4.658172)$ and then initialise the ReScaLE algorithm at a point one standard deviation away from $\hat{\beta}$ [R Core Team, 2017]. We transform our hazard function suitably (see Appendix (D)) and then allow the algorithm to run up to a diffusion time $t = 10^6$. A typical run of the ReScaLE algorithm together with the marginal densities of β_0 and β_1 are presented in the Figure (6.8). Figure (6.9) shows the comparison between the fit of the `glm()` estimate and the fit based on the estimate of the posterior mode of the ReScaLE algorithm. In Table (6.2) we present the efficiency and the uniform norm distance for the ReScaLE method with respect to Random-Walk Metropolis (RWM) algorithm and the Bernstein-von-Mises approximation, for 10 different runs. We used `MCMClogit()` function from the `MCMCpack` package in `R` [Martin et al.,

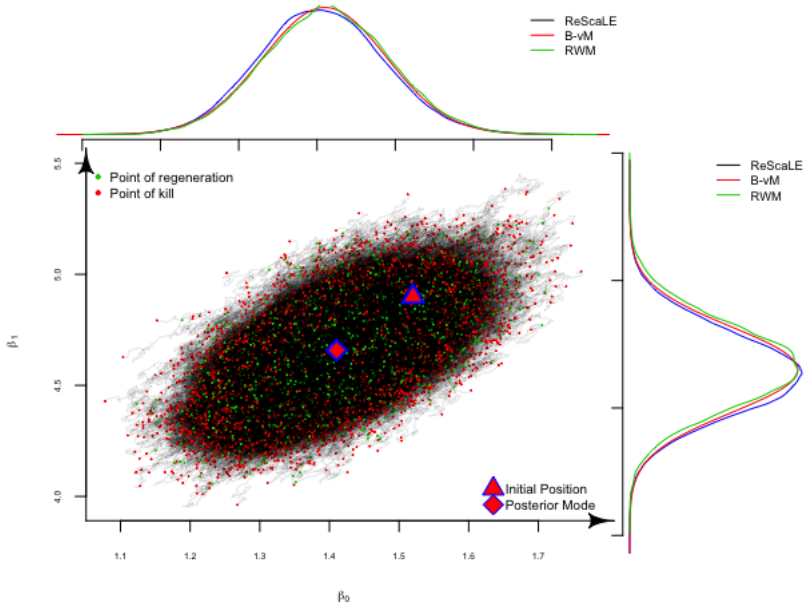


Figure 6.8: *ReScaLE for Menarche data: The central plot depicts the movement of the ReScaLE chain in the space of β. Points in green and red are the starting and the ending points of segments respectively. At the top and on the side we plot the marginal kernel density approximation of β_0 and β_1 respectively. The red curves are their respective Bernstein-von-Mises approximation. The curve in green is the kernel density estimate of a Random-Walk Metropolis run.*

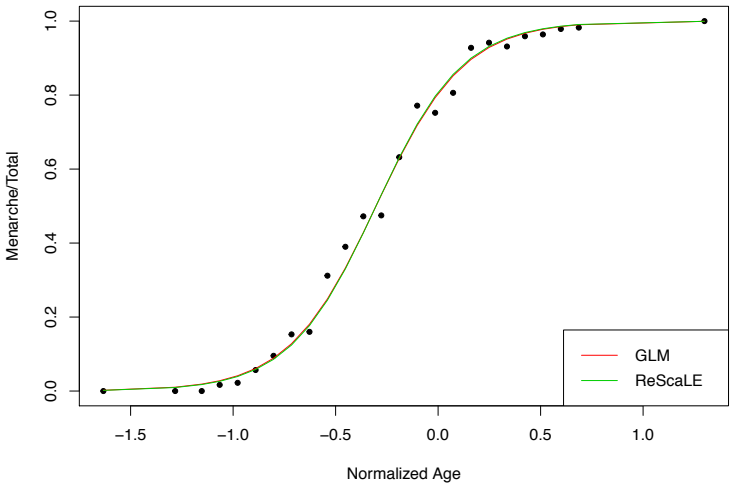


Figure 6.9: *ReScaLE for Menarche data: A comparison of the `glm()` and the ReScaLE fit to the `menarche` dataset.*

2011]. Furthermore, we estimated the potential scale reduction factors for β_0 and β_1, they are 1.000724 and 1.001572 respectively. This indicates a sufficient convergence to our posterior density. We present the evolution of the potential scale reduction factor as a function of the diffusion time in Figure (6.10).

	UND-RWM		UND-B-vM		Efficiency(ESS/sec)	
Run	β_0	β_1	β_0	β_1	β_0	β_1
1	0.0195	0.0147	0.0112	0.0305	55.6708	55.0341
2	0.0116	0.0102	0.0140	0.0466	58.7746	58.5114
3	0.0463	0.0734	0.0279	0.0310	56.9887	56.3025
4	0.0198	0.0196	0.0330	0.0590	57.3940	56.7321
5	0.0336	0.0345	0.0157	0.0110	57.0332	56.3525
6	0.0423	0.0541	0.0257	0.0132	59.8159	59.5666
7	0.0178	0.0250	0.0303	0.0201	65.3096	64.5350
8	0.0247	0.0521	0.0063	0.0103	66.0003	65.2446
9	0.0056	0.0130	0.0183	0.0346	66.1771	65.3875
10	0.0463	0.0734	0.0279	0.0310	65.6319	64.8890
Average	0.0267	0.0370	0.0210	0.0287	60.8796	60.2555

Table 6.2: *ReScaLE for Menarche data: This table presents the estimated uniform norm distance (2.5.2) of the ReScaLE algorithm (with respect to a Random-Walk Metropolis (RWM) and the Berntein-von-Mises (B-vM) method) and the efficiency (2.5.5) of $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1)$ for 10 different runs of the ReScaLE algorithm.*

6.2.3 Logistic Regression: Bioassay Experiment

In this example, we consider a small-sample where the posterior distribution is highly skewed. Here, our aim is to illustrate the working of the ReScaLE method to draw samples from a skewed posterior distribution. From an inferential point of view, this example illustrates that a Bernstein-von-Mises approximate provides an incorrect point estimate of the posterior mode in this setting. Furthermore, calculations of the posterior credible interval is also incorrect. The data set under consideration (called *bioassy* experiment data) comes from [Racine et al., 1986], which has also been used by [Gelman et al., 2008] to illustrate the Bayesian methodology.

Bioassay experiments are performed on animals to test the effect of various levels of dosage of a chemical compound on a group of animals. Typically, the group of animals are subjected to a level of drug dosage. The animal's response (for example dead/alive, cancerous/non-cancerous) is measured as a binary response variable.

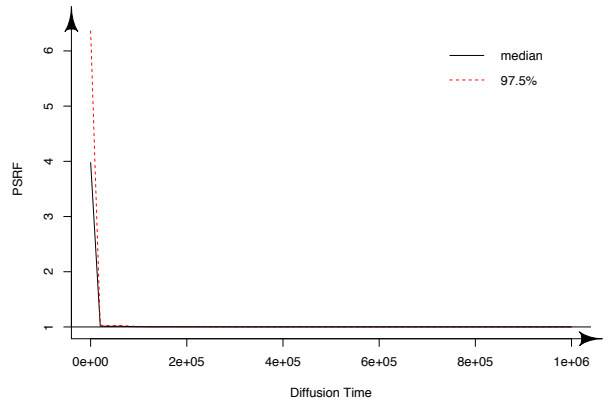


Figure 6.10: *ReScaLE for Menarche data: This figure illustrates the evolution of the potential scale reduction factor (2.5.9) of β_0 as a function of diffusion time t based on 10 runs of the ReScaLE algorithm. The solid line represents the median value of PSRF while the dotted line is the upper 97.5% confidence interval. The absence of fluctuation in this graph after $t \approx 2 \times 10^4$ shows that our method has converged to the target density.*

One such experiment considered in [Racine et al., 1986] is presented in Table (6.3) where dose level x_i is in the logarithm of g/ml.

Dose Level, x_i	No. of animals, N_i	No. of deaths, n_i
-0.86	5	0
-0.30	5	1
-0.05	5	3
0.73	5	5

Table 6.3: *Bioassay data considered in [Racine et al., 1986].*

In this case the aim is to sample according to the posterior distribution of a logistic regression, which predicts whether or not an animal dies given its dosage level. We use a logistic regression model similar to (6.2.4), (6.2.5) and (6.2.6) on the bioassay data presented in Table (6.3). Here we have transformed the dosage level (x_i) to have zero mean and a standard deviation of 0.5 [Gelman et al., 2008]. We use a Cauchy prior distribution with center 0 and scale 10 for parameter β_0 and a Cauchy distribution with center 0 and scale 2.5 for parameter β_1 [Gelman et al., 2008]. We transform the hazard function suitably (see Appendix (D)) and allow the ReScaLE algorithm to run up to a diffusion time $t = 5 \times 10^5$. A typical run of the ReScaLE algorithm together with the marginal densities of parameters β_0 and β_1 are presented in Figure (6.11). We also compare our ReScaLE run with respect to a Random-Walk Metropolis run (using `MCMClogit()` function in the `R`-package

`MCMCpack`). We can observe a clear discrepancy between the kernel density estimate of a ReScaLE run and the Bernstein-von-Mises approximation which implies that the latter is not a good approximation of the posterior density (Figure (6.11) on the right) in this case. Table (6.4) presents the estimated uniform norm distance for 10 different runs of the ReScaLE algorithm with respect to both the RWM and B-vM approximation. We can observe a clear discrepancy between the empirical distribution of the ReScaLE runs and the Bernstein-von-Mises approximation for the parameter β_1. This is because of the fact that the marginal posterior density of β_1 is highly skewed to the right. Interestingly, the efficiency of the ReScaLE algorithm drops down considerably as compared to the Menarche example (see Table (6.2)). For a skewed density the change in the first derivative is large which results in large values of $\kappa(\cdot)$ and therefore the efficiency decreases. Furthermore, we estimated the potential scale reduction factors for β_0 and β_1, which were 1.00521 and 1.00572 respectively. This indicates a sufficient convergence to our posterior density.

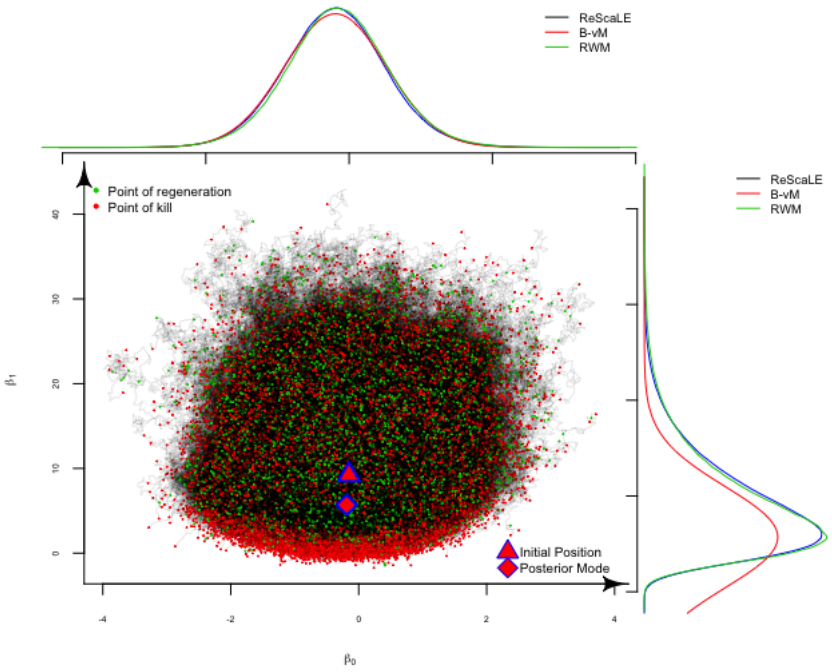


Figure 6.11: *ReScaLE for Bioassay data: The central plot depicts the movement of the ReScaLE chain in the space of β. Points in green and red are the starting and the ending points of segments respectively. At the top and on the side we plot the marginal kernel density approximation of β_0 and β_1 respectively. The red curves are their respective Bernstein-von-Mises approximation. The curve in green is the kernel density estimate of a Random-Walk Metropolis run.*

	UND-RWM		UND-B-vM		Efficiency(ESS/sec)	
Run	β_0	β_1	β_0	β_1	β_0	β_1
1	0.0208	0.0421	0.0457	0.2546	9.8360	9.8360
2	0.0403	0.0406	0.0610	0.2537	10.2522	10.2522
3	0.0057	0.0269	0.0202	0.2655	10.4218	10.4218
4	0.0415	0.0197	0.0657	0.2624	9.0129	9.0129
5	0.0780	0.0586	0.0969	0.2745	7.3570	7.3570
6	0.0174	0.0560	0.0421	0.2481	9.8316	9.8316
7	0.0299	0.0241	0.0186	0.2578	10.1274	10.1274
8	0.0172	0.0764	0.0340	0.2766	10.1087	10.1087
9	0.0189	0.0724	0.0292	0.2463	10.7781	10.7781
10	0.0223	0.0129	0.0105	0.2614	10.7000	10.7000
Average	0.0292	0.0430	0.0424	0.2601	9.8426	9.8426

Table 6.4: *ReScaLE for Bioassay data: This table presents the estimated uniform norm distance (2.5.2) of the ReScaLE algorithm (with respect to both a Random-Walk Metropolis (RWM) and the Bernstein-von-Mises (B-vM) method) and the efficiency (2.5.5) of* $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1)$ *for 10 different runs of the ReScaLE algorithm.*

6.3 Constructing Quasi-Stationary Monte Carlo Draws when the Hazard Rate is Unbounded

In Section (6.1) we outlined an approach for simulating a trajectory of a killed Brownian motion, where its hazard rate $\kappa(\cdot)$ is bounded (i.e $\exists\, K > 0$ such that $\kappa(\cdot) \in [0, K]$). In that context, we used Poisson thinning (5.2.1) to simulate the potential killing times of a Brownian motion. However, in situations where this bound is large, implementing Poisson thinning becomes impractical. Furthermore, in a general scenario the hazard rate might not be bounded globally and therefore, it is not possible to simulate the potential killing times of a Brownian motion directly.

We circumvent this issue using the localisation technique described in Section (5.2.2) (ref to Section (3.3) for more details) [Chen and Huang, 2013]. A user specifies a sequence of hypercubes in conjunction with the simulation of the first passage times of a Brownian motion. In our setting, a user-specified hypercube $\mathcal{H}_1$ allows us to bound the hazard rate for all points inscribed within it.

We now notice that we can find bounds for ϕ (similarly for κ). The continuity of the $\phi(\cdot)$ function is sufficient to make sure that the upper $U^1_{\mathbf{x}}$ and the lower $L^1_{\mathbf{x}}$ bound of $\phi(\cdot)$ defined by

$$L^1_{\mathbf{x}} := \inf_{\mathbf{X}_t \in \mathcal{H}_1} \phi(\mathbf{X}_t) \quad \text{and} \quad U^1_{\mathbf{x}} := \sup_{\mathbf{X}_t \in \mathcal{H}_1} \phi(\mathbf{X}_t) \tag{6.3.1}$$

can be evaluated for all points inside the hypercube (ref to Section (5.1) for assumptions). Hence

$$L^1_{\mathbf{x}} \leq \phi(\mathbf{X}_t) \leq U^1_{\mathbf{x}} \quad \text{for} \quad \mathbf{X}_t \in \mathcal{H}_1. \tag{6.3.2}$$

We can now simulate the potential death events $\{\xi_i\}_{i\geq 1}$ for our path, using the auxiliary Poisson process of rate $U^1_{\mathbf{x}} - \Phi$ (since $\kappa(\mathbf{X}_t) = \phi(\mathbf{X}_t) - \Phi \leq U^1_{\mathbf{x}} - \Phi$ whenever $L^1_{\mathbf{x}} \leq \phi(\mathbf{X}_t) \leq U^1_{\mathbf{x}}$) in the interval $[0, \tau_1]$. In this case, we identify the first instance of kill given by the killing time t_{kill} as:

$$t_{kill} := \min\{\xi_j \,:\, \xi_j > 0, U_j \leq \kappa(\mathbf{x}_{\xi_j})/(U^1_{\mathbf{x}} - \Phi), U_j \sim U[0,1]\}. \tag{6.3.3}$$

A user can now encounter two situations: either our Brownian motion leaves the hypercube $\mathcal{H}_1$ before time t, or the process is killed before exiting $\mathcal{H}_1$. If Brownian motion leaves the hypercube $\mathcal{H}_1$ before time t, we then construct a new hypercube $\mathcal{H}_2$ centred at the exit position $\mathbf{x}_{\tau_1}$ of the preceding hypercube $\mathcal{H}_1$. We then obtain new local bounds for $\phi(\mathbf{x})$ for points $\mathbf{x}$ in hypercube $\mathcal{H}_2$ and continue the simulation of potential kill times until the process either leaves the hypercube or is killed within it. In the latter case, when the process is killed within the hypercube, we employ an approach outlined in Chapter (4) for regenerating the process at the time when the hazard occurs i.e. t_{kill} [Blanchet et al., 2016]. As motivated in Section (6.1.2), we can regenerate the path by simulating the time of regeneration by $t_{reg} \sim U[0, t_{kill}]$. The regenerated position $\mathbf{x}_{reg}$ is simulated according to the law of a Brownian bridge between $(\xi_{k-1}, \mathbf{x}_{\xi_{k-1}})$ and $(\xi_k, \mathbf{x}_{\xi_k})$, whenever $\xi_{k-1} < t_{reg} < \xi_k$ for some k, for which a simulation mechanism has been discussed in Sections (3.5) and (3.6). An informal treatment of this idea has been discussed in Figure (6.12). Once our Brownian motion regenerates at the time of kill, we construct a new hypercube centred around $\mathbf{x}_{reg}$, evaluate the bounds for $\phi(\cdot)$ and continue in a similar manner as stated earlier. The above outlined procedure has been illustrated in the Figures (6.13) and (6.14) while its algorithmic details are presented in Algorithm (6.3.1).

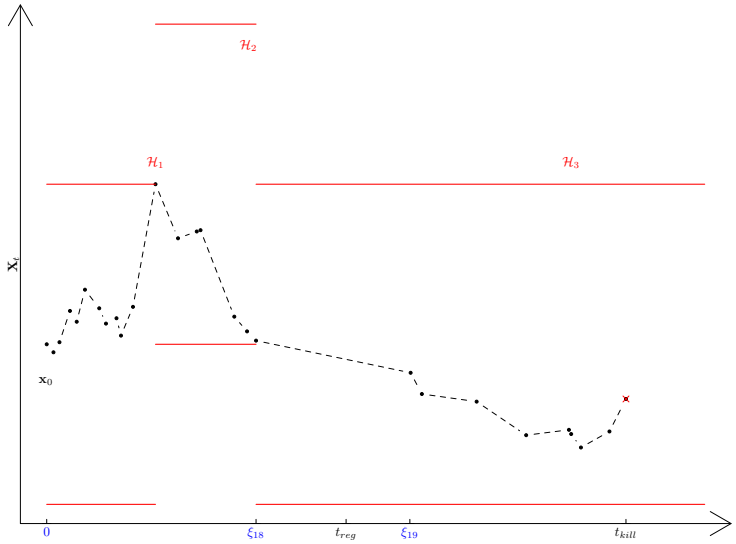


Figure 6.12: $\mathbf{x}_{reg}$ *is sampled by first simulating the time of regeneration* $t_{reg} \sim U[0, t_{kill}]$. *This is followed by sampling the position of a Brownian motion at* t_{reg} *conditional on the layer information* $(R^{(1)}_{\mathbf{x}}, R^{(2)}_{\mathbf{x}}, R^{(3)}_{\mathbf{x}})$ *and hypercubes* $(\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)$. *However, using the strong Markov property the position* $\mathbf{x}_{reg}$ *can be sampled using the law of a Brownian bridge process between times* ξ_{18} *and* ξ_{19} *conditioned on the layer information* $R^{(3)}_{\mathbf{x}}$ *which is constrained within the hypercube* $\mathcal{H}_3$.

Algorithm 6.3.1: ReScaLE Algorithm – Unbounded Hazard($\mathbf{X}_0, t, \mathbf{m}$)

input t : maximum time, $\mathbf{X}_0$: initial position, a time mesh $\mathbf{m} : m_1 < ... < m_n$
global κ, Φ
output ($\mathbf{X}_{m_1}, ..., \mathbf{X}_{m_n}$: positions of the killed Brownian motion.)
set $i \leftarrow 1, j \leftarrow 0, l \leftarrow 0, \xi_0 \leftarrow 0, \mathbf{X}_{\xi_0} \leftarrow \mathbf{X}_0, \mathbf{S} \leftarrow \emptyset$
while $\xi_j \leq t$
do
1. Create hypercube $\mathcal{H}_i$ centred at $\mathbf{X}_0$ and evaluate $U^i_{\mathbf{x}}$.
2. Simulate the layer information $R^{(i)}_{\mathbf{x}} = (\tau_i, \mathbf{x}_{\tau_i})$(Algorithm (3.4.2)) and set $\mathbf{S} = \mathbf{S} \cup (\tau_i, \mathbf{x}_{\tau_i}, R^{(i)}_{\mathbf{x}})$
3. KILL $\leftarrow$ **false**
while KILL $\neq$ **true**
do
4.a. Simulate $E \sim Exp(U^i_{\mathbf{x}} - \Phi)$
4.b. Set $j = j + 1$ and $\xi_j = \min\{\xi_{j-1} + E, \tau_i, t\}$
if $\xi_j = \tau_i$
do Set $i = i + 1, \mathbf{X}_0 = \mathbf{x}_{\tau_i}$ and return to step 1.
4.c. Simulate $\mathbf{X}_{\xi_j} \sim MVN(\mathbf{X}_{\xi_{j-1}}, (\xi_j - \xi_{j-1})I_d) \,|\, R^{(i)}_{\mathbf{x}}$ (ref to Section (3.5))
4.d. Calculate hazard probability $P = (\kappa(X_{\xi_j}))/(U^i_{\mathbf{x}} - \Phi)$ and simulate $U \sim U[0, 1]$.
do
4.e. Set $\mathbf{S} = \mathbf{S} \cup (\xi_j, \mathbf{X}_{\xi_j}, R^{(i)}_{\mathbf{x}})$
if $U \leq P$
then
5.a. KILL $\leftarrow$ **true** , $l \leftarrow l + 1$
5.b. Simulate regenerating time $t_{reg_{(l)}} \sim U[0, \xi_j]$ and evaluate k such that $\xi_{k-1} \leq t_{reg_{(l)}} \leq \xi_k$ and corresponding $R^{(\cdot)}_{\mathbf{x}}$.
5.c. Simulate $\mathbf{X}_{reg_{(l)}}$ using the law of Brownian bridge between times ξ_{k-1} and ξ_k conditional on $R^{(\cdot)}_{\mathbf{x}}$. (ref to Section (3.6))
5.d. Set $\mathbf{S} = \mathbf{S} \cup (t_{reg_{(l)}}, \mathbf{X}_{reg_{(l)}}, R^{(\cdot)}_{\mathbf{x}})$
5.e. Set $i = i + 1, \mathbf{X}_0 = \mathbf{X}_{reg_{(l)}}$ and return to step 1.
6. Evaluate Brownian bridge position at times $m_1, ..., m_n$ given $\mathbf{S}$
return ($\mathbf{X}_{m_1}, ..., \mathbf{X}_{m_n}$)

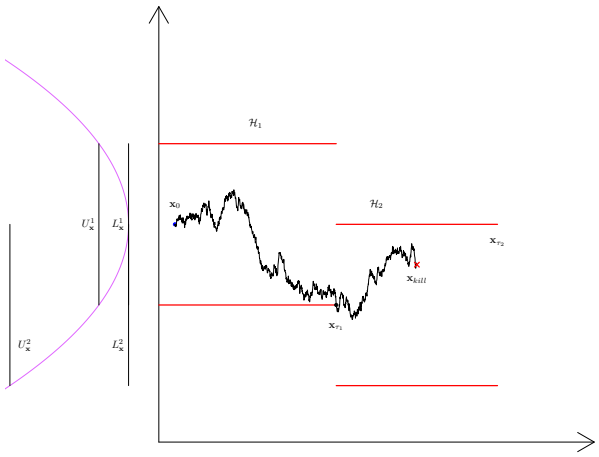


Figure 6.13: *This plot depicts the construction of hypercubes for constructing the sample path of a one-dimensional killed Brownian motion. For a typical function $\kappa(\cdot)$ (purple curve on the left), once the hypercube $\mathcal{H}_1$ is constructed, the layer information namely $R^{(1)}_{\mathbf{x}} = (\tau_1, \mathbf{x}_{\tau_1})$ is calculated. The hypercube is then used to find the upper $U^1_{\mathbf{x}}$ and lower $L^1_{\mathbf{x}}$ bounds of the function $\phi(\cdot)$ in order to sample the potential killing times of a Brownian motion. It can be further observed that our Brownian motion exits the hypercube $\mathcal{H}_1$ without getting killed. This leads to the construction of a hypercube $\mathcal{H}_2$ centred at exit position $\mathbf{x}_{\tau_1}$ of the previous hypercube $\mathcal{H}_1$. Next, new bounds $U^2_{\mathbf{x}}$ and $L^2_{\mathbf{x}}$ are evaluated for generating the potential killing events in the hypercube $\mathcal{H}_2$. Brownian motion is killed in the hypercube $\mathcal{H}_2$ at the position $\mathbf{x}_{kill}$.*

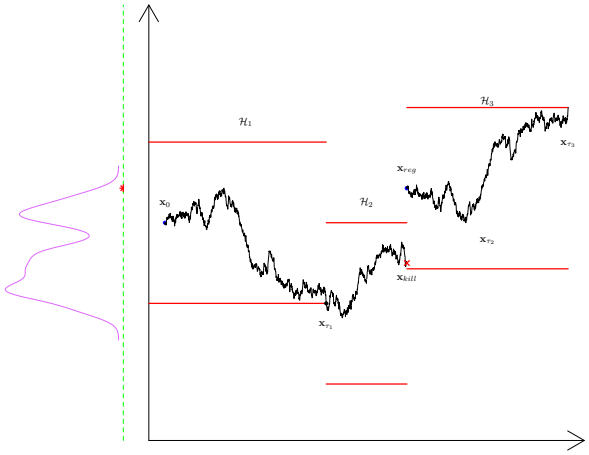


Figure 6.14: *This plot follows figure (6.13) and illustrates the instance once our Brownian motion is killed in the hypercube $\mathcal{H}_2$. Once the Brownian motion is killed, its trajectory is regenerated by sampling uniformly from the path visited by the process. This is depicted by drawing one sample from the empirical density of the path visited by our Brownian motion (empirical density depicted by the purple curve on the left). The generated sample $\mathbf{x}_{reg}$ becomes the starting point of Brownian path at the time of kill. This is followed by a construction of a hypercube $\mathcal{H}_3$ centred at the regeneration point $\mathbf{x}_{reg}$ and the layer information $R^{(3)}_{\mathbf{x}} = (\tau_3, \mathbf{x}_{\tau_3})$.*

6.4 Sub–Linear Cost of the ReScaLE Method

We pointed out in Section (5.3) that the computational cost involved in the calculation of the non-negative hazard rate $\kappa(\cdot) = \phi(\cdot) - \Phi$ is $\mathcal{O}(n)$, where we considered the entire data of size n. This results in an iterative algorithmic cost of $\mathcal{O}(n)$, which can be problematic in a tall data setting. However, we circumvented this issue by replacing $\kappa(\cdot)$ with an unbiased estimator $\tilde{\kappa}(\cdot)$, which also uses the control variate information. The computational cost of an unbiased estimator is not only cheaper but also guarantees convergence to the true posterior density [Pollock et al., 2017]. Similar to ScaLE, it allows the ReScaLE algorithm to achieve the scalability with respect to the data size n. Therefore, we employ the construction of unbiased estimators developed in Section (5.3) and replace the true hazard rate by its unbiased estimator. The construction of its bounds can be achieved in a manner outlined in Section (5.3.2) and Appendix (D), (D.1) . As a result the construction of the hypercube in Step (1.) of Algorithm (6.3.1) can be processed using newly constructed upper bound of an unbiased estimator (ref. to Section (5.3.2) for more details). For the remaining empirical studies in this chapter we use the scalable version of the ReScaLE algorithm in order to illustrate the scalability of our method, unless otherwise stated.

6.5 Experimental Studies - II

Here we present some applications of the ReScaLE algorithm developed earlier. At first, we consider an application of the ReScaLE algorithm to sample from the posterior distribution of a logistic regression model built on the US airline data. This example illustrates the working of the ReScaLE algorithm on a tall data problem. This empirical study is followed by an interesting case study of a logistic regression model on a synthetic data, where the posterior density assumes a non-Gaussian form. Finally, we present an empirical study of the efficiency of the ReScaLE algorithm with respect to varying data sizes n, to illustrate the scalability of the ReScaLE method. In this setting, we compare its performance against the Random-Walk Metropolis algorithm and the Zig-Zag method.

6.5.1 Bayesian Logistic Regression : The US Domestic Airline Data

The airline on-time performance data from the 2009 American Statistical Association Data Expo is used to demonstrate the working of the ReScaLE methodology on a tall data set. The data set is available at (`http://stat-computing.org/`

dataexpo/2009/the-data.html, http://euler.stat.yale.edu/~mjk56/Airline.tar.bz2) for public use and research purposes. This data set has been used previously by [Kane, 2010; Gunawan et al., 2017] and [Wang et al., 2016a] to demonstrate the scalable methodologies to work with big data. This data set constitutes the arrival and the departure information of all commercial flights operating within the US between October 1987 to April 2008. It consists of a record of $n = 120748239$ flights for 29 variables and when uncompressed, it requires 12 GB of RAM to be accessed directly into the memory.

We consider the problem of simulating from the posterior distribution of a logistic regression model where the response variable (y) determines whether or not the flight has arrived late. The Federal Aviation Administration (FAA) considers an arriving flight to be late when it reaches 15 minutes later than its scheduled arrival time. Therefore,

$$y_i = \begin{cases} 1 & \text{if the } i^{th} \text{ flight is delayed by more than 15 minutes,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.5.1}$$

We consider three covariates out of which two are binary and one is continuous [Gunawan et al., 2017]. The first binary covariate (X_1) is `weekend` which is defined as:

$$X_{1i} = \begin{cases} 1 & \text{if the } i^{th} \text{ flight operates on Saturdays/Sundays,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.5.2}$$

The second binary covariate (X_2) is `night` which is defined by

$$X_{2i} = \begin{cases} 1 & \text{if the departure time of } i^{th} \text{ flight is between 8PM and 5AM,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.5.3}$$

[Kane, 2010]. The remaining covariate (X_3) is `distance` which is the distance in 1000 miles from the origin to the destination. We normalize the distance variable by subtracting the minimum and dividing by the range. We then fit the logistic model of the form

$$y_i = \begin{cases} 1 & \text{with probability} \quad 1/(1+\exp\left(-\beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i} - \beta_3 X_{3i}\right)), \\ 0 & \text{otherwise.} \end{cases} \tag{6.5.4}$$

In order to run the scalable version of the ReScaLE algorithm (6.3.1), we use the control variate information namely $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3)$. A good candidate of $\hat{\beta}$ can be constructed using the `glm()` function in `R`, which roughly takes 6 minutes on a machine with 812 GB RAM and 48 cores. In this case we use a suitable transformation of our posterior density (see Appendix (D) and (C)) and then use our algorithm to obtain samples from it. We present the trace-plot and the marginal densities for each component of β for a typical run of the ReScaLE algorithm in Figure (6.15). For the logistic regression on the US domestic airline data, we use 10 independent runs of the ReScaLE method, each of them were allowed to run for a maximum time $t = 10^6$. The estimated uniform norm distance $\hat{d}_{tv}$ for each component is presented in Table (6.5). Finally, we present an estimate of the potential scale reduction factor in Table (6.7). Recall from Section (2.5) that these values indicate sufficient convergence to the target density.

	UND-B-vM				UND-RWM			
Run	β_0	β_1	β_2	β_3	β_0	β_1	β_2	β_3
1	0.0206	0.0139	0.0086	0.0146	0.0180	0.0143	0.0313	0.0217
2	0.0121	0.0171	0.0085	0.0167	0.0129	0.0210	0.0292	0.0192
3	0.0223	0.0106	0.0265	0.0301	0.0211	0.0159	0.0117	0.0407
4	0.0229	0.0283	0.0140	0.0118	0.0193	0.0321	0.0372	0.0235
5	0.0387	0.0445	0.0125	0.0191	0.0494	0.0523	0.0209	0.0236
6	0.0119	0.0127	0.0135	0.0136	0.0158	0.0162	0.0291	0.0239
7	0.0159	0.0180	0.0187	0.0070	0.0152	0.0245	0.0446	0.0214
8	0.0130	0.0102	0.0082	0.0141	0.0264	0.0154	0.0303	0.0203
9	0.0158	0.0180	0.0187	0.0069	0.0151	0.0246	0.0445	0.0214
10	0.0200	0.0191	0.0108	0.0189	0.0179	0.0227	0.0371	0.0252
Average	0.0193	0.0192	0.0140	0.0153	0.0211	0.0239	0.0316	0.0241

Table 6.5: *ReScaLE for US airline data: This table presents the estimated uniform norm distance (UND) (2.5.2) between the empirical distribution of ReScaLE runs with respect to the Bernstein-von-Mises approximation (B-vM) and a Ramdom-Walk Metropolis (RWM) run for each component of* $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3)$.

6.5.2 A Rare Event Logistic Regression

Earlier in Section (6.5.1) we encountered a situation where the posterior density takes a Gaussian form (approximated by the Bernstein-von-Mises approximation).

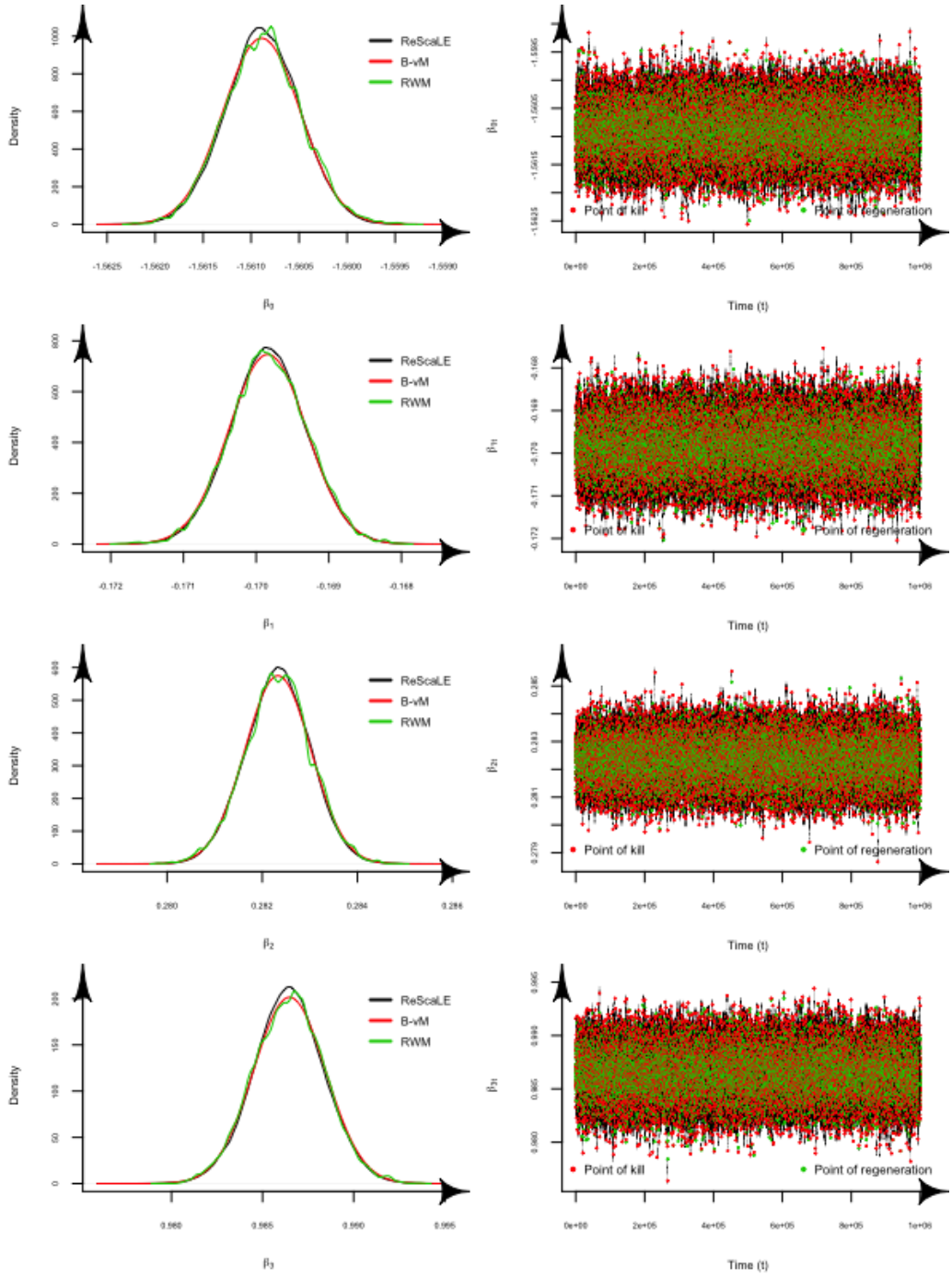


Figure 6.15: *ReScaLE for US airline data: This figure shows the kernel density approximation of the output obtained from the ReScaLE method together with its Bernstein von-Mises approximation (in red), for each coefficient of the logistic model on the US domestic airline data. On the right we present the trace plot for each coefficient. Visually it can be observed that the ReScaLE method approximates the posterior density very well.*

Run	β_0	β_1	β_2	β_3
1	13.7136	13.3744	13.3900	13.1488
2	12.4018	12.2287	12.5733	12.4608
3	13.3291	13.3139	13.5721	13.6545
4	12.4649	12.8741	13.0269	12.7272
5	13.6397	13.4347	13.7804	13.5503
6	12.5654	12.9359	12.0945	12.9469
7	11.0065	12.8731	11.8722	12.7122
8	12.9019	12.9247	13.3342	12.5225
9	12.1094	12.7593	12.5614	12.5403
10	13.1725	13.3444	13.0365	13.5273
Average	12.7305	13.0063	12.9241	12.9791

Table 6.6: *ReScaLE for US airline data: This table presents the estimated effective sample size (2.5.5) of each componenet of $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3)$. The table consists of the efficiency for 10 different runs of the ReScaLE method.*

	β_0	β_1	β_2	β_3
PSRF	1.003147	1.001787	1.004139	1.003199

Table 6.7: *ReScaLE for US airline data: This table presents the estimated value of PSRF (2.5.9) of each componenet of $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3)$. The PSRF has been estimated based on 10 different runs of the ReScaLE method.*

In this section we consider an example where the shape of the posterior density takes a non-Gaussian form in a tall data setting. As a result the Bernstein-von-Mises approximation cannot be used as a proxy for the posterior density. To illustrate this, we build a logistic regression model with two parameters, where the true values of parameters are set to $\beta_0 = -12.7$ and $\beta_1 = 1$ respectively. The covariate X_1 was generated using a normal distribution with mean 0 and variance 1. We simulated a data set of size $n = 10^6$, where the binary dependent variable y was generated using

$$u_i \sim U[0,1] \quad \text{and } y_i = \begin{cases} 1 & \text{if} \quad u_i \leq 1/(1+\exp(-\beta_0 - \beta_1 X_{1i})), \\ 0 & \text{otherwise,} \end{cases} \tag{6.5.5}$$

which resulted in 8 successes for the dependent variable y (ref to Appendix (E) for `R` codes). We present the bivariate trace plot of β together with their marginal densities in Figure (6.16). It can be observed that the marginal density of the parameter β_0 is skewed to the left and therefore, the Bernstein-von-Mises approximation (green curve at the top) is not a good estimator of the posterior density. In Figure (6.16), we compare the marginal density of a ReScaLE run against the marginal density of a RWM run. We use the `R` function `MCMCpack::MCMClogit()` to obtain the run of a

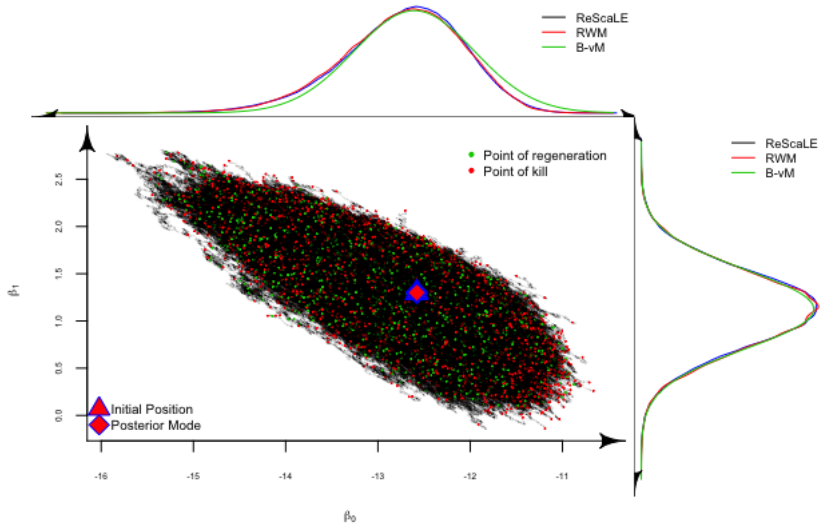


Figure 6.16: *ReScaLE for rare event logistic regression: This figure shows the bivariate trace plot of parameters (β_0, β_1) together with their respective marginal densities. The red curve is the marginal density resulting due to a run of the Random Walk Metropolis algorithm. The green curve is the Bernstein-von-Mises normal approximation. For a typical run of ReScaLE run the uniform norm distance between the empirical densities for the parameter β_0 are 0.01064868 (w.r.t RWM run) and 0.05815253 (w.r.t. Bernstein-von-Mises approximation).*

RWM algorithm (see for instance [Brooks et al., 2011, Chapter 1], [Sherlock et al., 2010] and [Martin et al., 2011]). Although the marginal densities of the two methods are indistinguishable, the average efficiency (see Section (2.5)) of the ReScaLE method (25.8 based on 10 runs) is significantly higher than a RWM algorithm (0.92). Furthermore, the estimate of the potential scale reduction factor for β_0 and β_1 are given by 1.019334 and 1.01316 respectively. This signifies that the ReScaLE chain explores the support of the posterior density very well. Therefore, it is evident that the ReScaLE method exhibits the scalability in this setting, which we explore more in Section (6.5.3).

6.5.3 Efficiency of the ReScaLE Method w.r.t the Data Size

In this section we examine the efficiency (see Section (2.5)) of the ReScaLE method with respect to varying data sizes n. This is followed by a comparison of the efficiency with respect to a Random Walk Metropolis (see for instance [Brooks et al., 2011, Chapter 1] and [Sherlock et al., 2010]) and the *Zig-Zag* algorithm [Bierkens et al., 2016]. To achieve this, we consider a logistic model with two parameters where the true values of the parameters were set to $\beta_0 = -0.5$ and $\beta_1 = 1$ respectively.

The only covariate was generated using a normal random variable with mean 0 and variance 1. `RZigZag` package was used to run the Zig-Zag method on the logistic model considered here [Bierkens et al., 2016, R-package]. For computational reasons, we considered data sizes up to 10^9 for the ReScaLE method. However RWM and ZigZag could only be run until data sizes of 10^6 and 10^7 respectively. The efficiency of the Zig-Zag method could not be calculated for data sizes larger than 10^7 due to computational issues involved in the `RZigZag` package.

The three methods considered here were allowed to run until diffusion time $t = 10^6$ which guaranteed desired level of convergence. We should note that, for the Zig-Zag and the ReScaLE method, the computational cost involved in the calculation of the control variate information has been discounted. This requires an $\mathcal{O}(n)$ computation at the start of the algorithm but it needs to be calculated only at the start of the algorithm. In the case of the ReScaLE method the $\mathcal{O}(n)$ control-variate information was computed using the `glm()` function in `R`. The `ZigZag` method uses Newton's method to find the control-variate information which is also discounted for in our analysis [Bierkens et al., 2016].

A comparison of the efficiencies with respect to varying data sizes are presented in Figure (6.17). It can be observed from the plot that the efficiency for the smaller data sizes is higher for the RWM algorithm. In this case, the efficiency of the two methods is roughly equal for the data size 5×10^4. However, for the data sizes larger than 5×10^4, the ReScaLE method wins over the RWM algorithm. Therefore, it is wise to use RWM for smaller data sizes but for a substantially large data set it can be computationally cumbersome. Figure (6.18) shows the relative efficiency of three methods with respect to the initial efficiency when 5 data points are considered in the logistic regression model. It illustrates that the Zig-Zag method and RWM depreciates in efficiency as the data size increases, while the ReScaLE method remains fractionally more efficient than the other two. This exhibits the scaling property of the ReScaLE method, showing that the efficiency of our method remains roughly constant with an increasing data size. Furthermore, we plot the number of individual likelihood computed per unit diffusion time in Figure (6.19). It can be observed that the number of individual likelihood per unit diffusion time roughly remains constant with respect to the data size for the ReScaLE algorithm however, for the other two methods it increases with an increase in the data size. This also suggests that the computational complexity of the ReScaLE method remains sub-linear with respect to the data size. It is interesting to note that when the computational cost of

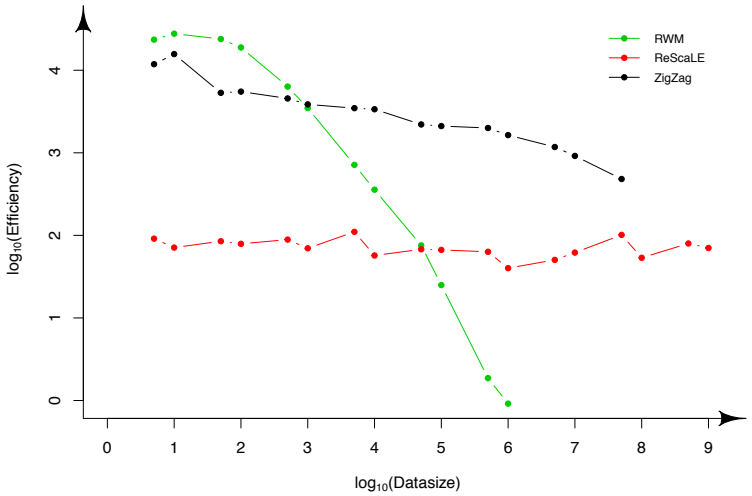


Figure 6.17: *This figure illustrates the efficiencies (2.5.5) of the ReScaLE method (red curve), Zig-Zag (black curve) and the Random-Walk Metropolis algorithm (green curve) with respect to the data size, for the intercept parameter β_0. Contrary to the ReScaLE method, the other two methods degrade in efficiency as the data size increases.*

control-variate is included, the overall cost of ReScaLE and ZigZag becomes $\mathcal{O}(n)$. However, slopes of the computational cost for ReScaLE and ZigZag would remain below the RWM method. This is because of the fact that each iteration of RWM is $\mathcal{O}(n)$ however for the other two methods computational cost of each iteration is $\mathcal{O}(1)$.

6.5.4 Behaviour of the ReScaLE Method under Varying Sub-Sampling Size

In this section we systematically study the behaviour of the ReScaLE method with respect to a sub-sampling size. We consider two parameter logistic regression examples for data size 10^5 where the true values of the parameters were set to $\beta_0 = -0.5$ and $\beta_1 = 1$ respectively. The only covariate was generated using a normal random variable with mean 0 and variance 1. We allow the ReScaLE method to run until a desired level of convergence is achieved. We perform 5 different runs of our algorithm and study the computational time and efficiency of our method for various levels of sub-sampling sizes. Figures (6.20) and(6.21) illustrate the computational time and efficiency (2.5.5) plotted against an increasing level of sub-sampling sizes within this example. Figure (6.22) shows the number of likelihood evaluations per second as a

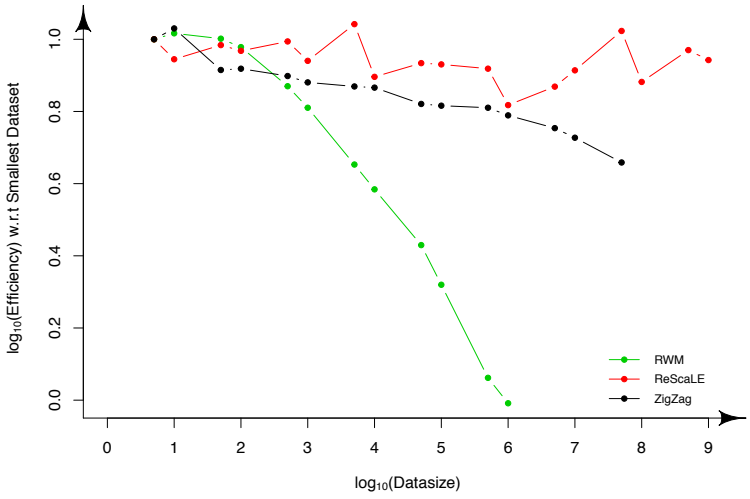


Figure 6.18: *This figure illustrates the fraction of efficiency (2.5.5) with respect to the case when 5 data points are considered in the logistic regression model. A decreasing sloped line signifies that the method becomes less efficient with an increasing data size. This empirical study illustrates that the ReScaLE algorithm roughly remains more efficient as compared to the two other methods considered here.*

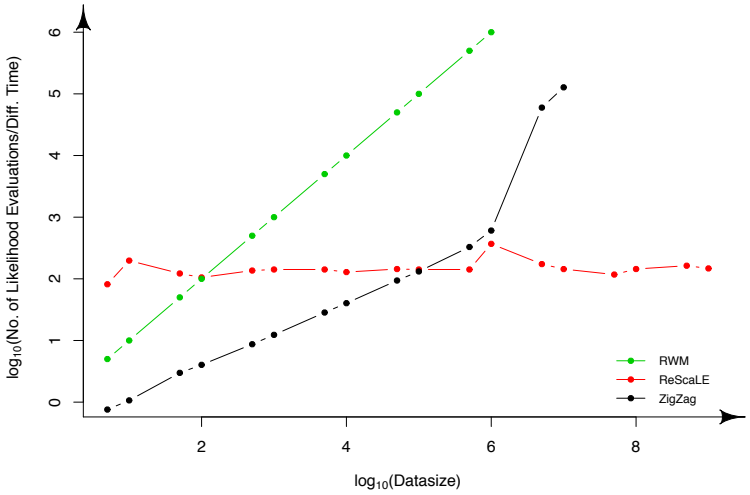


Figure 6.19: *This figure illustrates the number of likelihood calculations per unit of diffusion time for various algorithms considered. The ReScaLE algorithm possesses a zero sloped line however, the other two methods have an increasingly sloped line.*

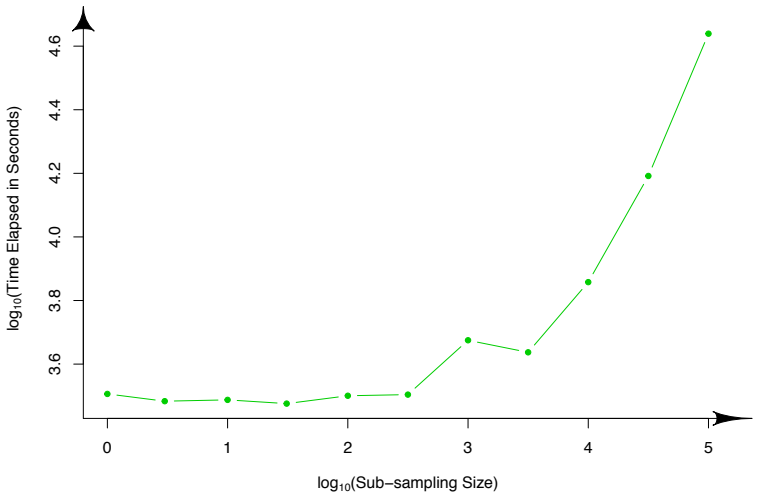


Figure 6.20: *This figure illustrates the computational time with respect to the various levels of sub-sampling sizes considered in the logistic regression model.*

function of the sub-sampling size, which clearly posses an increasing linear trend. We can further observe that the computational time increases exponentially with an increase in the sub- sampling size (Figure (6.20)) as a result. The efficiency of the ReScaLE method decreases exponentially with an increase in the sub-sampling size (Figure (6.21)). This inference suggest that a lower value of a sub-sampling size (relative to data size) should be used within the big data setting to reduce the computational bottleneck.

6.5.5 Efficiency of ReScaLE w.r.t. Dimensions

Let us recall from Section (5.4) that the computational cost of a QSMC algorithm is dependent on the upper bound of the hazard rate. This bound increases at least linearly as a function of the dimensionality of a problem (ref to expression (5.3.13)). This results in an increased number of simulations of the first passage times and the potential killing times. This is mainly because $\tau^{(1)} > ... > \tau^{(d)}$, where $\tau^{(d)}$ denotes the first passage time for a d–dimensional Brownian motion exiting a hypercube of unit layer size (ref to Section (3.4.2)). Therefore, for a unit time interval, the average number of simulated first passage times increases with an increase in the number of dimensions, although its rate of increase is unknown.

We empirically investigate the above motivation using a toy simulation problem.

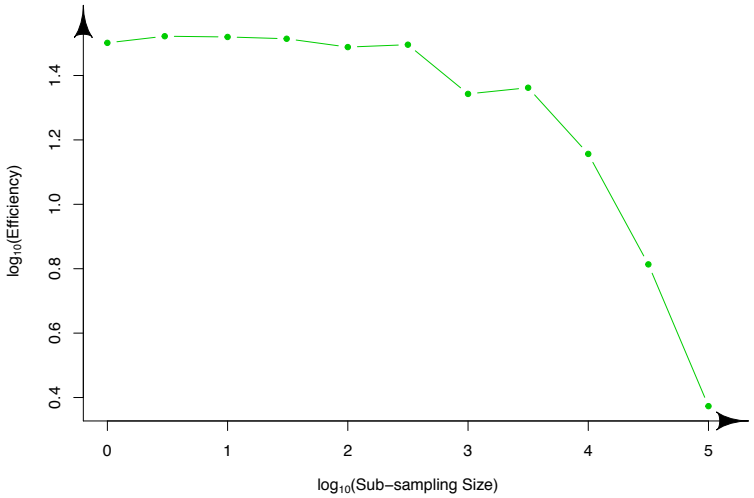


Figure 6.21: *This figure illustrates the efficiency with respect to the various levels of sub-sampling sizes considered in the logistic regression model.*

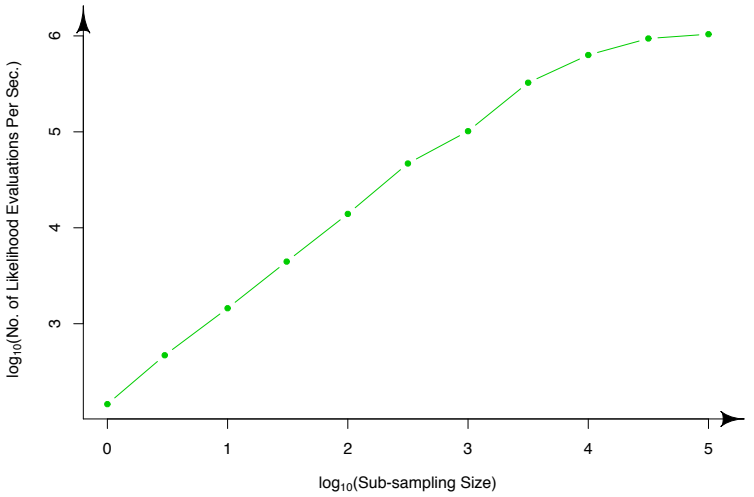
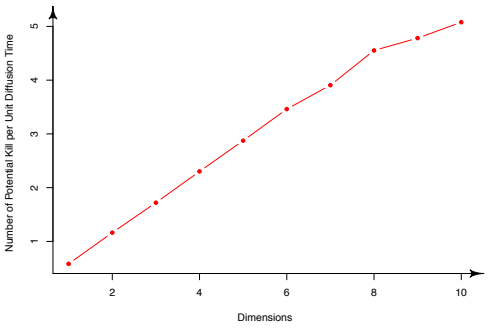
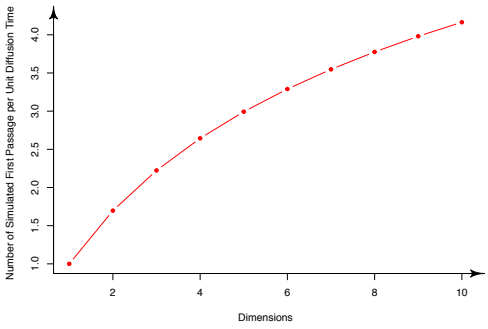


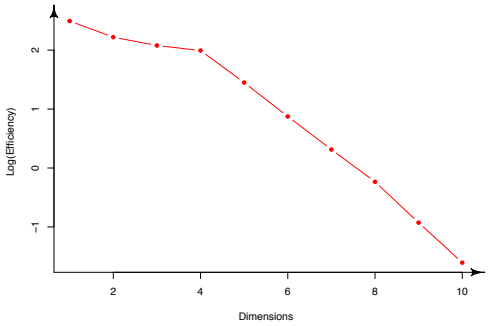
Figure 6.22: *This figure illustrates the number of likelihood evaluations with respect to various levels of sub-sampling sizes considered in the logistic regression model.*



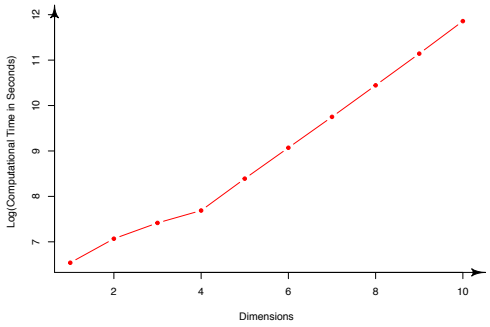
(a) *The average number of potential killing events per unit of diffusion time.*



(b) *The average number of first passage times per unit of diffusion time.*



(c) *Logarithm of the efficiency* (2.5.5).



(d) *Logarithm of computational cost in seconds.*

Figure 6.23: *ReScaLE for $N(\mathbf{0}, I_d)$ simulation: Here the ReScaLE algorithm was run for a maximum diffusion time of $t = 10^4$.*

Here ,we are interested in drawing samples according to $N(\mathbf{0}, I_d)$ for varying values of d. For each dimension d, we obtain 5 runs of our algorithm. We should note from Figure (6.23a) that the average number of potential killing events increases roughly linearly as a function of dimension. However, the number of first passage times has a non-linear trend as a function of dimensions. This supports our non-linear trend of the computational cost in Figure (6.23d). We present the efficiency of the ReScaLE method plotted against dimensions in Figure (6.23c). We can observe that the overall efficiency of our algorithm roughly decreases exponentially as a function of the dimensionality.

6.6 Discussion

We have devised a new QSMC algorithm which uses the mathematical foundations developed in [Pollock et al., 2017] and targets the quasi-stationary density using the regenerative approach of [Blanchet et al., 2016]. This chapter illustrated the construction of our QSMC algorithm for two cases; one when the hazard rate was bounded and secondly when it was unbounded. The mathematical form of the hazard rate allowed us to employ sub-sampling within it and as a result we achieved $\mathcal{O}(1)$ iterative computational cost with respect to the data size. We witnessed various applications of the newly constructed QSMC algorithm to both real and synthetic data. When we invoked sub-sampling, we were able to obtain samples from the posterior distribution of a tall data problem. This can be computationally infeasible when one considers a traditional MCMC method to draw samples according to a tall data posterior. Compared to the existing QSMC algorithm ScaLE, our new algorithm provides an edge in terms of the computational cost. This can be intuitively understood from the use of a single trajectory within the ReScaLE algorithm as opposed to a large number of particles in the former case.

While our QSMC algorithm is encouraging in a tall data problem, it faces a major hurdle especially when a multi-modal posterior is under consideration. It can struggle to converge when modes are separated far apart. This problem can be intuitively explained in our case. When a Brownian motion tries to travel from one mode to the other, it gets instantaneously killed in the region of high hazard between the two modes. Therefore, our Brownian motion is forced to over-explore the mode it is currently in. The 'strong memory' of the regenerative approach [Blanchet et al., 2016] for quasi-stationarity forces our algorithm to be trapped in the current mode. These characteristics ultimately result in a lack of convergence. However, we should note that the multi-modal characteristics of a posterior is rare in a tall data setting. We explore this situation in detail in the next Chapter (7).

Our QSMC algorithm can be sensitive to its initial point, especially a poor start (when the algorithm is initialised far from the posterior mode), which can force the algorithm into persistence issues. This situation is of particular importance when a user incorrectly observes the posterior mode. A 'strong memory' (recall that Algorithm (4.1.1) regenerates according to its entire history) of the regenerative approach does not allow our algorithm to forget the bad start it had. As a result our algorithm keeps revisiting the poor region and hence resulting into persistence

issues. We address these problems in greater detail in Chapter (7).

We can observe from Chapter (5) that the trajectories of the ScaLE algorithm possesses the Markov property. Unlike ScaLE, the evolution of a ReScaLE trajectory is dependent on its behaviour in the past and hence a ReScaLE trajectory exhibits non-Markovian characteristics. During the course of the ScaLE algorithm, a user can therefore discard the initial part of the trajectories which over-represents the unlikely part of the state-space. However, this is not possible within the ReScaLE algorithm due to its non-Markovian nature, which requires us to regenerate uniformly according to the entire history of a chain. Especially under a poor start where the ReScaLE algorithm initially explores the unlikely part of the state-space; this leads to a computational bottleneck within the ReScaLE algorithm as a user is required to store its entire trajectory.

A user can find herself in a situation where some likelihood components provide much more 'information' to the ϕ function than the other. This situation is of particular relevance when a few rare observations are under consideration. In this context the likelihood terms corresponding to rare observations would provide significantly more information to our posterior than the others. When sub-sampling is invoked, these likelihood terms are picked rarely and hence the sub-sampling can be ineffective. To capture these rare pieces of information a user can increase the sub-sampling size within the ReScaLE algorithm to enhance the chances of corresponding data points being sampled. However, the increase in the sub-sampling size results in a reduced efficiency of our algorithm. From a mathematical point of view, inclusion of more observations within the unbiased estimation of ϕ does not change its bounds for a given hypercube. Therefore, our underlying algorithm remains the same except for the fact that its computational cost increases for a fixed diffusion time, which results in a lower efficiency.

As far as a high-dimensional posterior distribution is concerned, like many MCMC methods, our method is not fully protected from the curse-of-dimensionality. We have observed though a toy problem that the ReScaLE algorithm loses its efficiency roughly exponentially as a function of dimensions. This however opens a new direction of research with a possibility of enhancing the efficiency of a QSMC algorithm, for a high-dimensional posterior distribution. One possible approach of addressing this issue is to perform sub-sampling on dimensions during the course of a QSMC algorithm [Pollock et al., 2017] albeit, it is unknown whether such a technique can

provide $\mathcal{O}(1)$ iterative computational cost as a function of dimensions, while achieving convergence at the same time.

The algorithmic structure of the QSMC algorithms proposed so far are highly sequential in nature. To put this informally, it is difficult to design a parallel architecture for these algorithms. This can usually become problematic when the data is too big to fit into a single memory of a computer. As a result the application of our algorithm to such data becomes impractical since it requires a ready access to all data points under consideration. However, the *Consensus Monte Carlo* type approach motivates us to run our favourite QSMC algorithm on the clusters of data in parallel and the posterior estimates are then 'suitably recombined'. While this approach would provide a greater processing speed, it is likely to come at the cost of exactness of our QSMC algorithm. Furthermore, the 'recombination' step remains a challenge in the *Consensus Monte Carlo* type methods which work mostly for Gaussian distributions.

Chapter 7

New Regeneration Strategies

In chapter (6) a uniform regeneration strategy motivated by Algorithm (4.1.1) was used when the process was killed. More formally, upon killing we chose to regenerate according to the empirical density of the states visited by the process. Within the ReScaLE algorithm however, a uniform regeneration strategy faces several issues as we discuss in Section (7.1) and (7.3). The empirical results presented in this chapter illustrate the computational inefficiencies faced by a uniform rebirth strategy. We further provide a non-uniform regeneration strategy which is able to circumvent the problem faced by its uniform counterpart and hence, illustrate that it is not only able to 'beat' its uniform counterpart but achieve faster convergence.

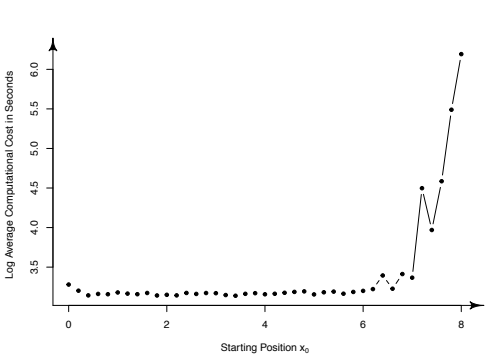
This chapter is organised as follows: Section (7.1) outlines some key issues faced by the uniform regeneration strategy. We present some empirical results in the context of the experiments considered in Chapter (6). This is followed by the construction of a non-uniform rebirth strategy in Section (7.2). We explore a series of findings to support the construction of our non-uniform rebirth strategy. We experiment with these newly constructed rebirth strategies on the Menarche and the US domestic airline data considered in Chapter (6). Finally, Section (7.3) proposes a head start regeneration strategy where a user can opt to regenerate according to the density of her choice initially. We motivate the use of a head start strategy through a toy example, where the ReScaLE algorithm shows a poor mixing property under the uniform rebirth strategy. However, we demonstrate that a meaningful density to regenerate within the ReScaLE algorithm can be chosen to circumvent the problem of a poor mixing.

7.1 A Poor Start of the ReScaLE Algorithm

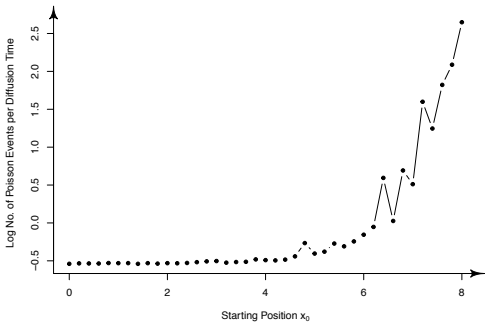
A poor start is considered problematic for most MCMC methods; this issue usually delays the convergence of the underlying MCMC method (see for instance [Raftery and Lewis, 1995; Owen et al., 2014]). Sometimes a poorly initialized method might not retrieve the correct target distribution at all [Gelman and Shirley, 2011]. A poor start is more problematic within the ReScaLE algorithm since it maintains a strong memory of the state-space visited in the past. As a result a poor start is more often followed by a poor regeneration, which is elaborated more in the following paragraphs. Therefore, we address the problem of a poor start in the context of our ReScaLE algorithm. We have observed earlier in Chapter (6) that the construction of the ReScaLE algorithm depends upon the control variate information. The control-variates inform the user about the location of the target mode. Often, only partial information is available to the user. In such contexts, an arbitrarily initialized ReScaLE algorithm will most likely have a poor start and hence, struggle to converge to the correct target distribution.

It should be noted that under the regenerative mechanism of [Blanchet et al., 2016], it might take longer to explore the entire support of a target distribution if the ReScaLE algorithm experiences a poor start, (see Figure (7.2a) and (7.3a) for instance). This is mainly due to the fact that the ReScaLE chain would regenerate more often from a poor region, given that it experiences a poor start (ref to Figure (7.2c) and (7.3c)for an example). Recall that this situation arises since the algorithm needs to regenerate from the occupation measure of the states visited by the process so far. Therefore, a poor start is more often followed by a poor regeneration, since the empirical weight of the poor region will be significant (Figure (7.2c) and (7.3c)). A trace plot has been depicted in Figure (7.4) in the context of the Menarche data; here, the ReScaLE algorithm is initialized at a point three standard deviations away from the posterior mode $\hat{\beta}$. We observe that a poor start in this situation leads to a persistent behaviour of the ReScaLE algorithm.

Under a poor start the ReScaLE algorithm simulates a higher number of death events, which is due to the fact that the hazard rate is high in the tails (illustrated in Figure (7.1b) and (7.5)). The ReScaLE trajectories at such death events are most often killed (see Figure (7.2d) for instance), however, the uniform rebirth strategy forces the algorithm to revisit the poor region (see Figure (7.2c) and (7.3c)). The problems above not only lead to a slower convergence, but at the same time make



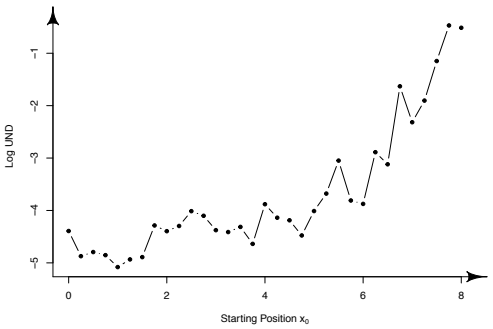
(a) *Average computational cost in seconds*



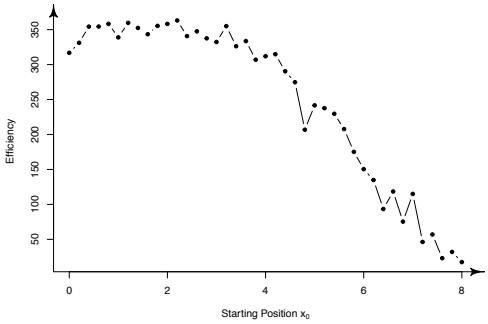
(b) *Average number of Poisson events per unit of diffusion time*

Figure 7.1: *ReScaLE for Standard normal target: This experiment is based on 5 runs of the ReScaLE algorithm for a standard normal target where maximum diffusion time* $t = 10^4$*. This figure illustrates the computational complexity as a function of the starting point. When the ReScaLE algorithm is initialised far off in the tails, both its computational cost and the number of proposed Poisson events increase significantly.*

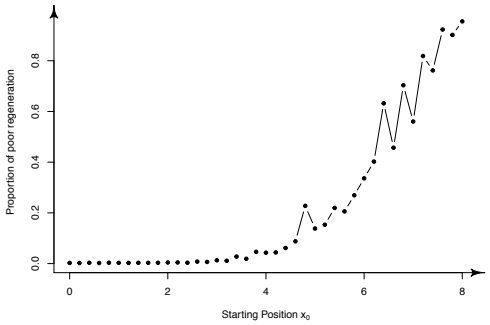
the algorithm computationally inefficient (see Figure (7.2b) and (7.3b) for instance). These problems have been demonstrated for a standard normal target (in Figure (7.1) and (7.2)) and the menarche data (Figure (7.3)). Therefore, we address the problems above using a different rebirth strategy outlined in the following Section (7.2).



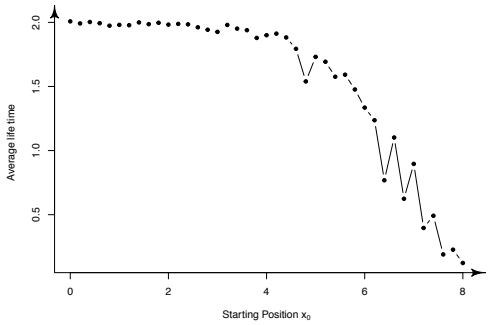
(a) *Logarithm of Uniform Norm Distance* (2.5.2)



(b) *Efficiency–Effective sample size per second* (2.5.5)

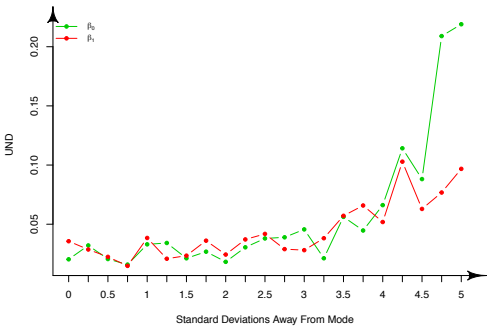


(c) *Proportion of poor regenerations: in this context it is given by the ratio of the number of regenerations in* $(-\infty, -3) \cup (3, \infty)$ *with respect to the total.*

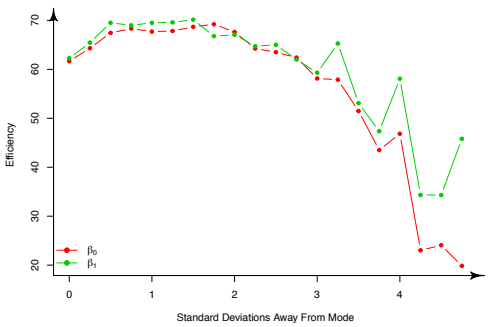


(d) *Average life span of segments: this is given by the average diffusion time elapsed between the start and kill of a given segment. See* (4.1.25) *for its definition.*

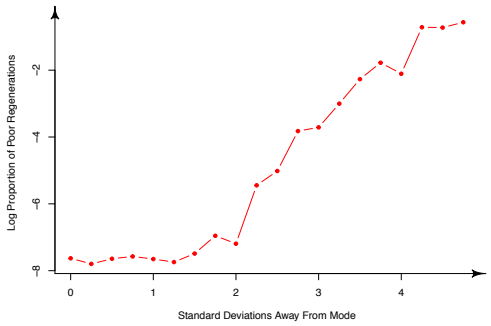
Figure 7.2: *ReScaLE for Standard normal target: Figure* (7.2a) *and* (7.2b) *illustrates the average of logarithm of the UND (between the kernel density estimate and the true density) and efficiency based on 5 runs of the ReScaLE algorithm for a standard normal target; here the maximum diffusion time* $t = 10^4$*. The further away we start from the mode, greater is the discrepancy between the target and its approximated density* (7.2a)*. The efficiency of the ReScaLE algorithm has a downward trend as a function of its starting value away from the mode* (7.2b)*. Figure* (7.2c) *demonstrates the percentage of times that a regeneration occurs in* $(-\infty, -3) \cup (3, \infty)$ *which we define as a poor region. It is clear that a poor start forces the algorithm to inherit a poor regeneration. Figure* (7.2d) *shows the average life span of a ReScaLE segment. The average life span follows a decreasing trend as a function of the distance of the starting value from the mode.*



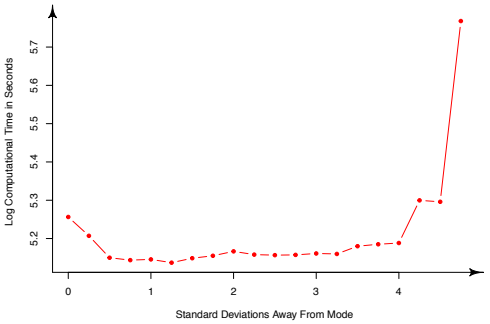
(a) *Uniform Norm Distance* (2.5.2)



(b) *Efficiency* (2.5.5)



(c) *Proportion of poor regenerations: in this context it is given by the percentage of the number of regenerations outside* 3 *standard deviations from the posterior mode.*



(d) *Average Computational time in seconds.*

Figure 7.3: *ReScaLE for Menarche Data: Figure* (7.3a) *and* (7.3b) *illustrates the average UND and efficiency based on 5 runs of the ReScaLE algorithm. The efficiency of the ReScaLE algorithm has a downward trend as a function of its starting value away from the posterior mode* (7.3b)*. Figure* (7.3c) *demonstrates the percentage of time that a poor regeneration occurs. We define a poor region as the part of state space which is outside* 3 *standard deviations away from the posterior mode. It is clear that a poor start forces the algorithm to inherit a poor regeneration. Figure* (7.3d) *shows the computational cost as a function of starting value.*

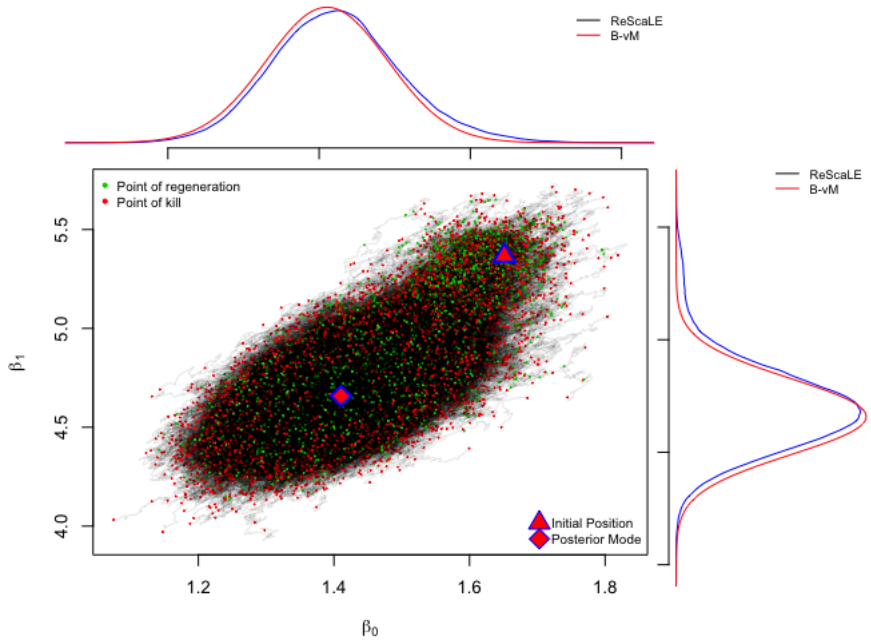


Figure 7.4: *ReScaLE for Menarche data: The central plot depicts the movement of the ReScaLE chain in the space of β, where it is initialised at a point 3 standard deviations away from $\hat{\beta}$. Points in green and red are the regenerations and the kill points of segments respectively. At the top and on the side we plot the marginal kernel density approximation of β_0 and β_1 respectively. The red curves are their respective Bernstein-von-Mises approximations.*

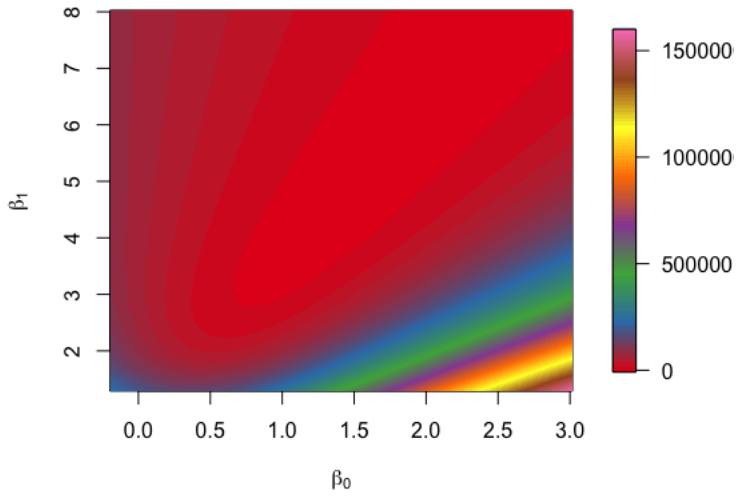


Figure 7.5: *ReScaLE for Menarche data: This plot illustrates the contour of the hazard function $\kappa(\cdot)$ as a function of $\beta = (\beta_0, \beta_1)$.*

7.2 A Non-Uniform Rebirth Strategy

In this section, we address the problem of the persistent behaviour resulting due to a poor start within in the ReScaLE algorithm. Recall that in Algorithm (6.1.1) and (6.3.1) a killed Brownian motion regenerates at the instance of kill, t_{kill}, by first simulating the time of regeneration, $t_{reg} \sim U[0, t_{kill}]$, followed by its position at t_{reg}. However, for larger values of t_{kill} it is natural to assume that samples obtained 'more recently' will be drawn from its quasi-stationary distribution. This is because a segment given by ReScaLE would have explored the quasi-stationary states before being killed at t_{kill}. A natural question therefore arises: can we regenerate from the recent past in lieu of the entire history? More formally, we hypothesize to regenerate $t_{reg} \sim U[\lambda \cdot t_{kill}, t_{kill}]$ for some $0 \leq \lambda < 1$, in order to achieve faster convergence and resolve the persistence issues as observed earlier. The time regeneration in this manner allows the ReScaLE algorithm to forget the experience of a poor start and hence it decreases the computational bottleneck. However, a choice of $\lambda \approx 1$ is discouraged, since at $\lambda \cdot t_{kill}$ our chain would have left the quasi-stationary states.

Algorithm (7.2.1) illustrates a modified version of (4.1.1), which takes into account our newly hypothesized regeneration strategy. However, we should note that the mathematical underpinning of Algorithm (7.2.1) has not been developed in this thesis. A recent work by Wang et al. [2018] constructs a time regeneration using a beta distribution where its mathematical foundations has been discussed in the context of the ReScaLE algorithm. Intuitively, similar to our λ–rebirth strategy a beta distribution is chosen such that it puts more weight to the recent history of the chain [Wang et al., 2018]. We use the natural extension of Algorithm (7.2.1) within the algorithmic construction of the ReScaLE method, which has been explained in Section (6.3). Here we assume the quasi-stationarity of a λ–rebirth strategy in our case. A typical run of the ReScaLE algorithm for a positive value of λ is presented in Figure (7.6). Unlike Figure (7.4), it can be observed that the ReScaLE algorithm is able to recover quickly even after being subjected to a poor start.

Algorithm 7.2.1: Simulating from a QSD: λ–Rebirth(π_0, t, λ)

output ($\mathbf{S}$: Skeleton of a given chain)

1. Initialize the probability vector $\tilde{\pi} = \pi_0$ on the non-absorbing states $\mathcal{T}$.
2. Set $s = 0$, $\mathbf{S} \leftarrow \emptyset$ simulate $X_s \sim \tilde{\pi}$ and set $\mathbf{S} \leftarrow \mathbf{S} \cup X_s$.

while $s < t$

do
- **while** Segment is not absorbed
  - **do**
    3. Set $\tilde{s} = s + ds$, simulate $X_{\tilde{s}}$ according to the law of $X_{\tilde{s}} \,|\, X_s$.
    4. Update $\mathbf{S} \leftarrow \mathbf{S} \cup X_{\tilde{s}}$.
    5. Update $\tilde{\pi}$ using $\tilde{\pi}_j = \dfrac{1}{(1-\lambda)\tilde{s}} \displaystyle\int_{\lambda \cdot \tilde{s}}^{\tilde{s}} \mathbb{I}(X_q = j) dq$ for all $j \in \mathcal{T}$

    until absorption.
- 6. Once absorbed at $\tilde{s}$, re-simulate $X_{\tilde{s}} \sim \tilde{\pi}$ and set $s = \tilde{s}$.

return ($\mathbf{S}$)

7.2.1 Menarche Data: λ–Rebirth Strategy Under a Poor Start

In this section, we study the performance of our algorithm under the non-uniform rebirth strategy, for varying values of λ. For the purpose of illustration we employ Algorithm (6.3.1) for a logistic regression model on the Menarche data, however, we do not invoke sub-sampling on $\kappa(\cdot)$ within this example. Here we examine whether or not a suitable value of λ can be found which leads to an optimal performance. To achieve this, we employ 10 different runs of the ReScaLE algorithm for various values of $\lambda \in [0, 1)$. For a logistic regression model on this data, the ReScaLE algorithm was initialised at a point three standard deviations away from the posterior mode. A typical run of the ReScaLE algorithm for a series of λ values has been plotted in Figure (7.7). It should be noticed that the recovery of the ReScaLE algorithm from a poor start enhances with an increase in the value of λ.

In this example, our aim is to find a value of λ which is able to outperform others, in terms of the performance measures described in Section (2.5). Therefore, we perform a systematic study of the uniform norm distance and $\tau_{und}(\epsilon)$ in Figure (7.8) and (7.9) respectively. We can observe that the uniform norm distance and $\tau_{und}(\epsilon)$ decrease until $\lambda \approx 0.75$ and from that point it starts to increase again. The plot of $\tau_{und}(\epsilon)$ in Figure (7.9) indicates that the speed of convergence enhances with an

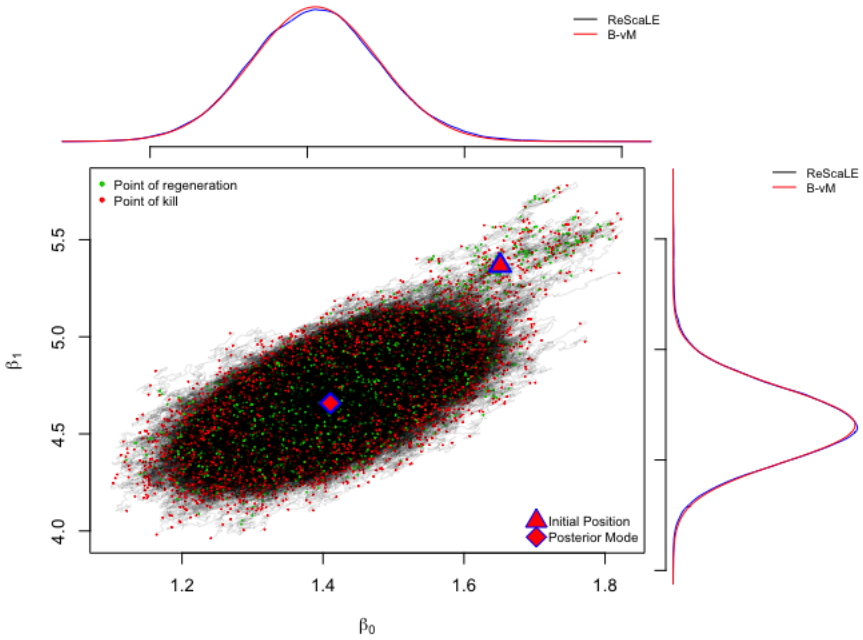


Figure 7.6: *ReScaLE for Menarche data: The central plot depicts the movement of the ReScaLE chain in the space of β, where the ReScaLE algorithm is initialised at a point 3 standard deviations away from $\hat{\beta}$ and λ was set at a value of* 0.5*. Points in green and red are the regenerations and the kill points of segments respectively. At the top and on the side we plot the marginal kernel density approximation of β_0 and β_1 respectively. The red curves are their respective Bernstein-von-Mises approximations.*

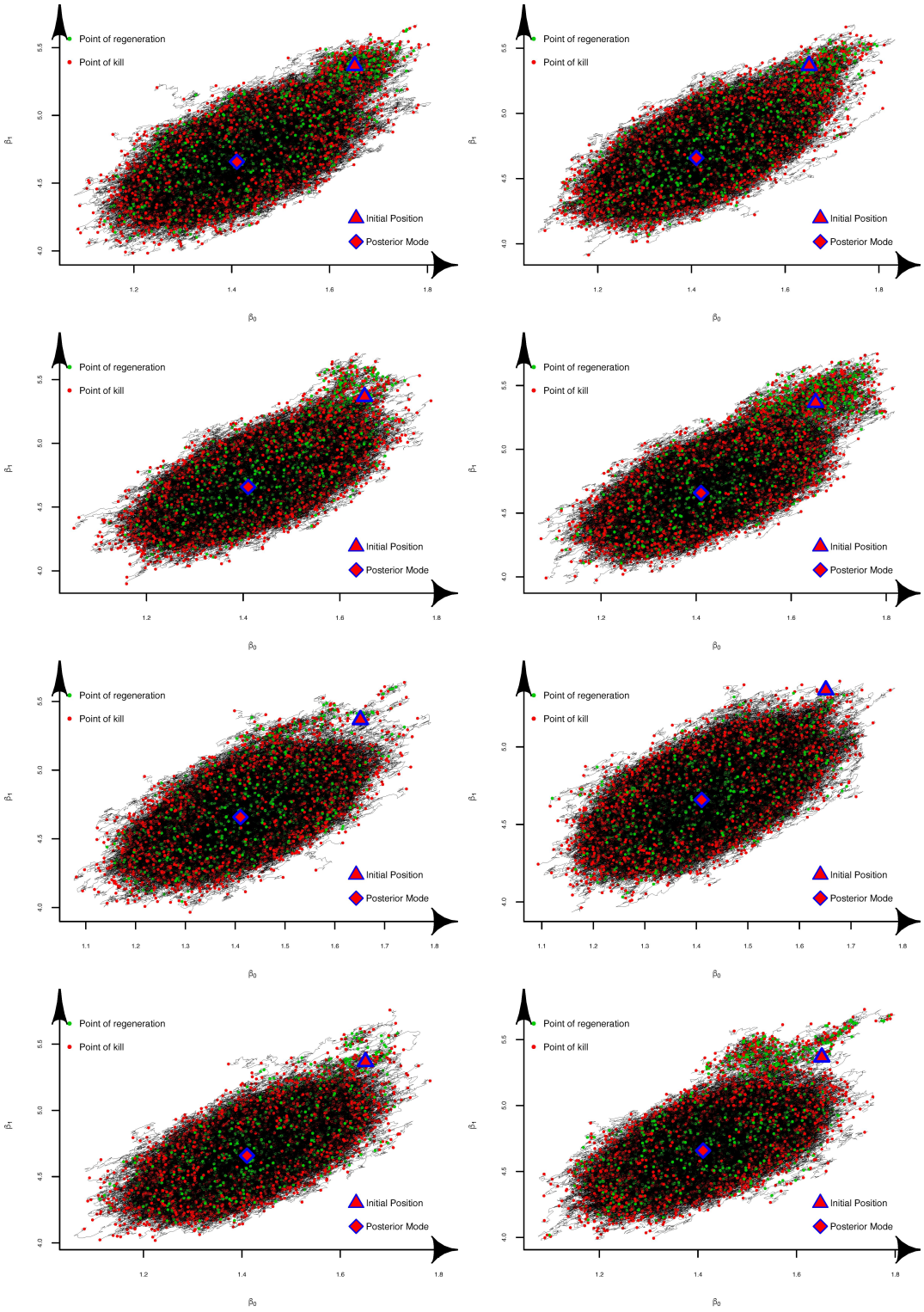


Figure 7.7: *ReScaLE for Menarche data: This figure illustrates the bivariate trace plot of β for $\lambda = 0, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.75$ (top left to bottom right).*

increase in the value of λ until $\lambda \approx 0.75$ and for $\lambda \approx 1$ it degrades in terms of the speed. Figure (7.10) illustrates the computational cost incurred against the values of λ. A clear downward trend suggests to use a higher value of λ to reduce the computational bottleneck. However, we observed from Figures (7.8) and (7.9) that a value of $\lambda \approx 1$ is suboptimal. Therefore, based on these findings it can be inferred that the optimal regeneration strategy is obtained for a λ close to 0.75. Furthermore, when we consider the mixing properties of the ReScaLE chain along with its computational cost i.e. efficiency, the optimal rebirth strategy corresponds to $\lambda \approx 0.9$; the result is presented in Figure (7.11). In this case we have assumed that our λ–rebirth strategy attains quasi-stationarity for the efficiency-based measures to be reasonable.

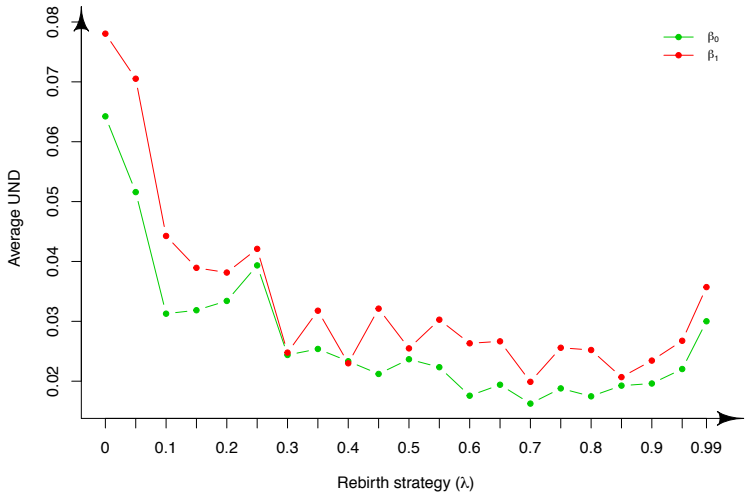


Figure 7.8: *ReScaLE for Menarche data: This plot illustrates the average uniform norm (based on 10 runs) distance (2.5.2) against various values of λ. It can be observed that the minimum value is achieved for λ close to* 0.70.

7.2.2 The US Airline Data: λ–Rebirth Strategy Under a Poor Start

We illustrate the working of a λ–rebirth strategy for the logistic regression model on the US airline data, where we invoke sub-sampling within our algorithm. For a given value of λ, we obtain 10 independent runs of the ReScaLE algorithm using a λ–rebirth strategy, for a maximum time of $t = 10^6$. We subjected each run of the ReScaLE algorithm to a poor start, where the algorithm was initialised at a point three standard deviations away from the posterior mode. The trace plot of the first component β_0 for a typical run of the ReScaLE algorithm has been presented in

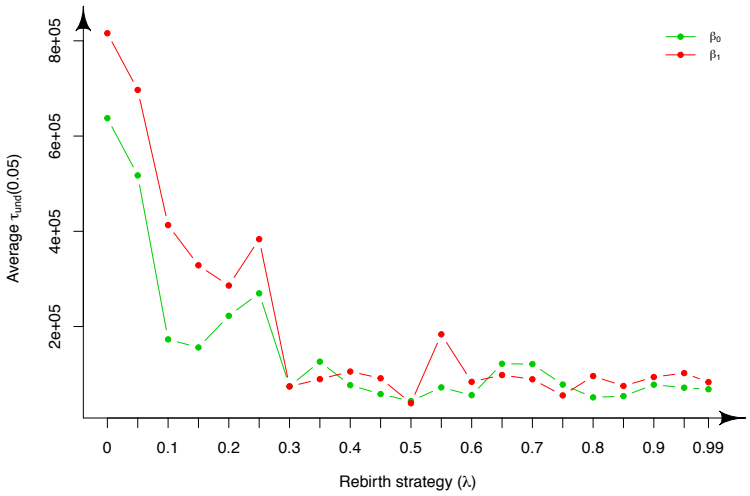


Figure 7.9: *ReScaLE for Menarche data: This plot illustrates the variation of* $\tau_{und}(\epsilon = 0.05)$ (2.5.3) *(based on 10 runs) against various values of* λ *for the Menarche data. It can be observed that the minimum value is reached at* $\lambda \approx 0.50$ *in this example.*

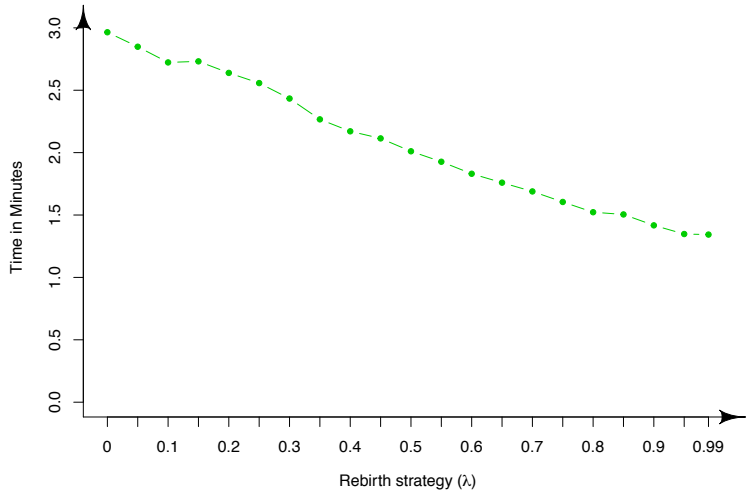


Figure 7.10: *ReScaLE for Menarche data: This plot depicts the average computational cost (based on 10 independent runs) incurred while using a* λ*– rebirth strategy on the Menarche data.*

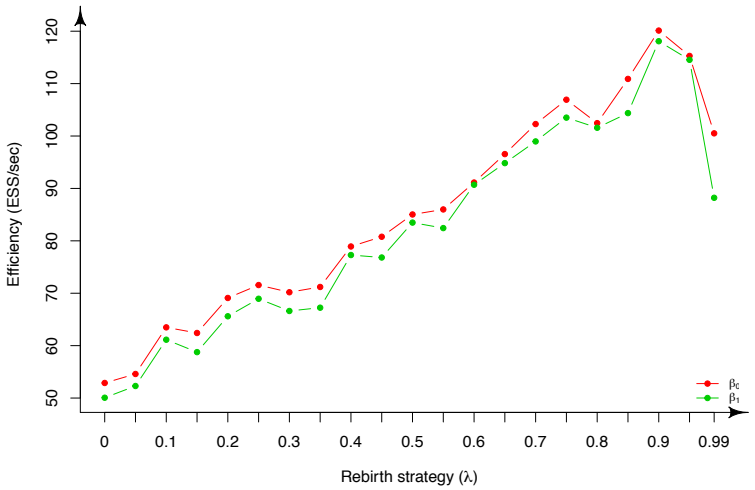


Figure 7.11: *ReScaLE for Menarche data: This figure illustrates the behaviour of effective sample size per unit of computational cost* (2.5.5) *across varying values of* λ.

Figure (7.12). With an increase in the value of λ it can be noticed that the chain is able to recover quickly from a poor start.

In Figure (7.13) we plot the average value of $\tau_{und}(0.05)$ of the ReScaLE algorithm using 10 different runs. It can be observed that a λ– rebirth strategy $(0 < \lambda < 1)$ outperforms its uniform counterpart (recall that $\lambda = 0$ corresponds to the uniform rebirth strategy of Blanchet et al. [2016]).

We can observe a clear decreasing trend for the performance measure $\tau_{und}(0.05)$ in Figure (7.13), which is followed by an increasing trend for λ close to 1. In this context, it can be concluded that an optimal choice of the regeneration parameter is λ close to 0.9. Also, when we consider the mixing properties of different rebirth strategies, together with their computational cost, this optimal rebirth strategy turns out to be $\lambda \approx 0.9$; the result is presented in the Figure (7.14).

7.2.3 Performance of λ–Rebirth Strategy Under Various Starts

Motivated by the success of our λ–rebirth strategy under a fixed poor initialisation, we analyse its performance when the ReScaLE algorithm is subjected to different starts. Here we consider different starting values of our algorithm for a sequence of $\lambda \in [0, 1)$. For each combination of λ and the starting position we obtain several

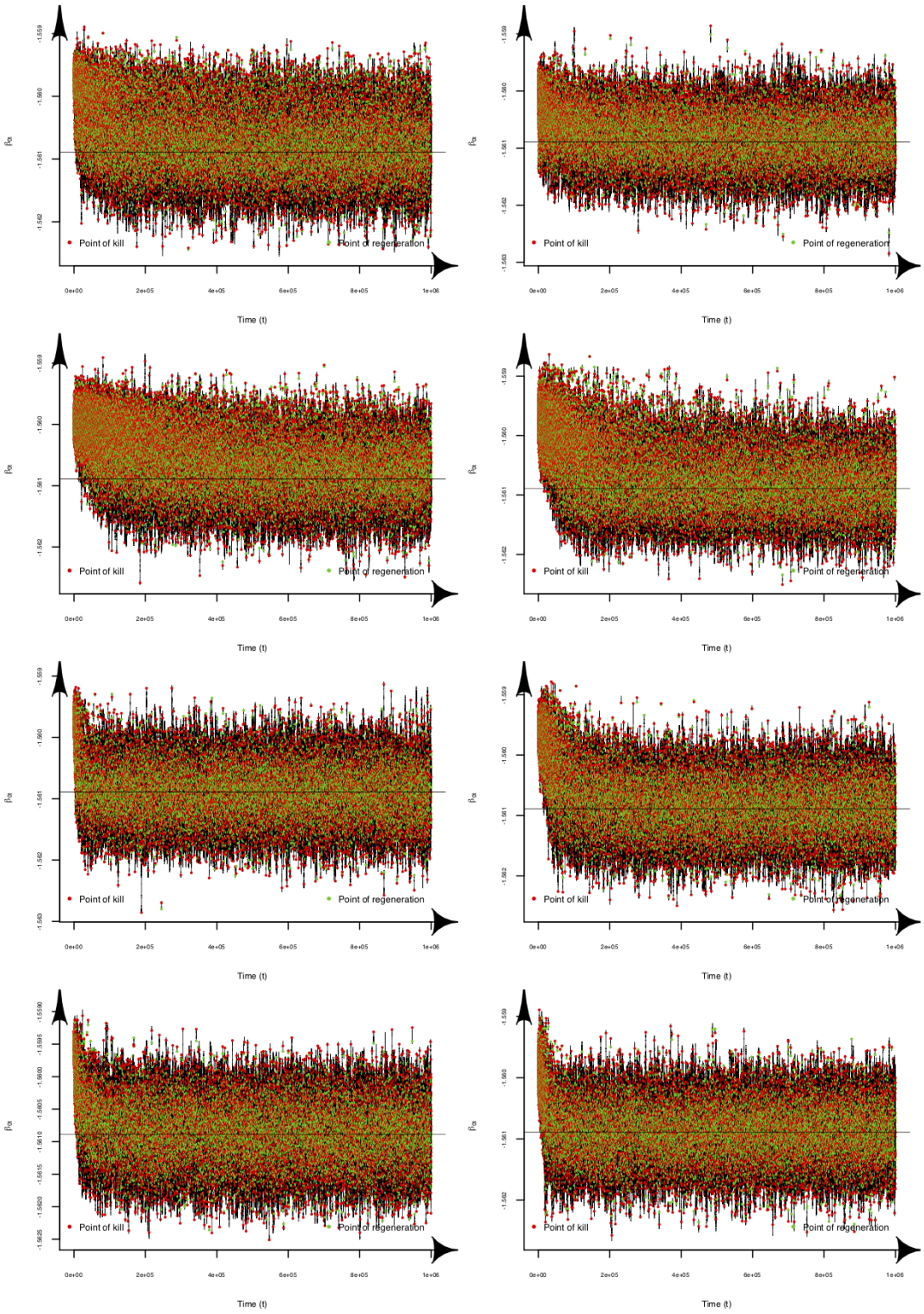


Figure 7.12: *ReScaLE for US airline data: This figure illustrates the trace plot of* β_0 *for* $\lambda = 0, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35$ *(top left to bottom right). The black horizontal line is the posterior mode of* β_0*.*

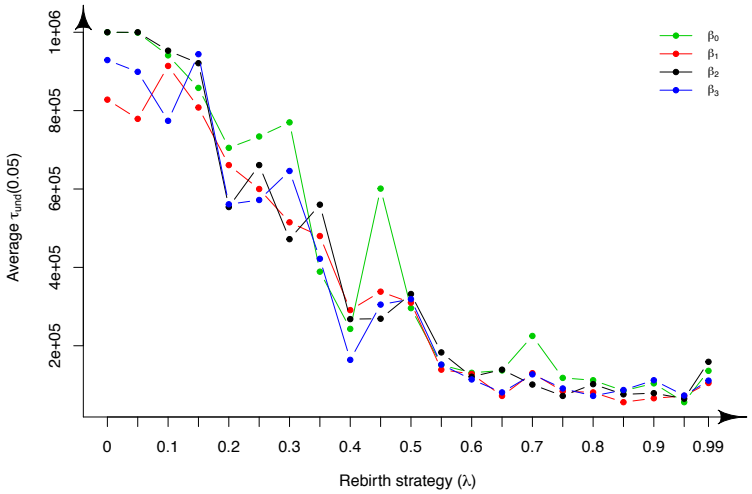


Figure 7.13: *ReScaLE for US airline data: This figure illustrates the average* $\tau_{und}(0.05)$ (2.5.3) *for the parameters of the logistic regression on the airline data. It can be observed that the overall minimum value is attained for* $\lambda \approx 0.90$.

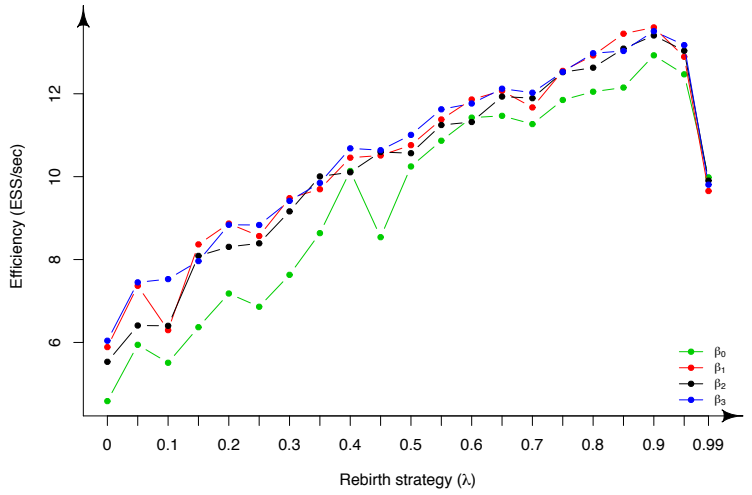
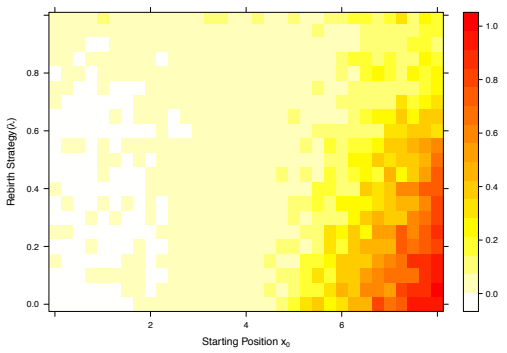


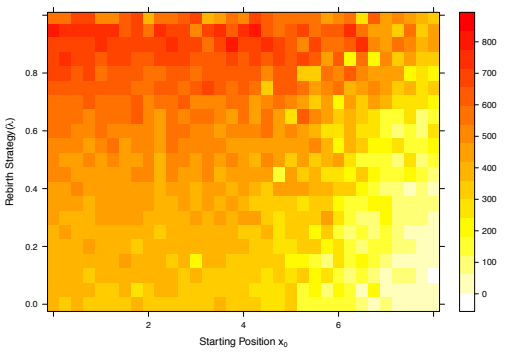
Figure 7.14: *ReScaLE for US airline data: This figure illustrates the behaviour of effective sample size per unit of computational cost* (2.5.5) *across varying values of* λ.

runs of the ReScaLE algorithm. In this case the experiment was performed on a standard normal target and the menarche data.

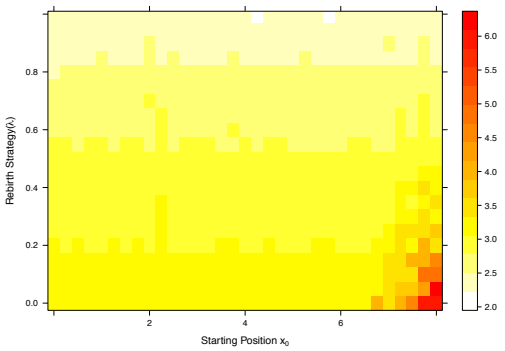
We should first note that the computational cost is reduced when we employ $\lambda > 0.5$ (see Figure (7.15c) and (7.16a)). An invocation of a λ–rebirth strategy allows our algorithm to forget a poor start and allows it to converge faster to the target density (see Figure (7.15a) and (7.15d) for instance). A λ–rebirth strategy improves the overall efficiency of our algorithm. Interestingly, a value of $\lambda \in (0.5, 0.95)$ provides a higher efficiency to our algorithm (see Figure (7.16b) and (7.15b)). However, we should note in each cases that $\lambda \approx 1$ is suboptimal. These findings suggest that one can employ a value of $\lambda \in (0.5, 0.95)$ within the ReScaLE algorithm irrespective of its initial position.



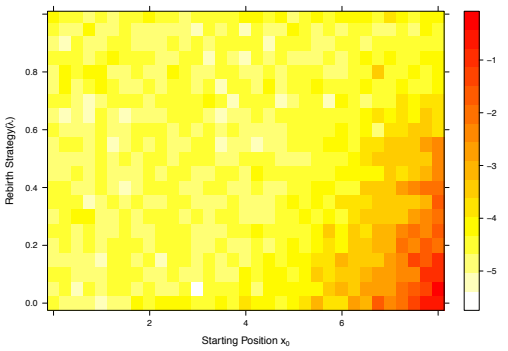
(a) *Proportion of poor regeneration for every combination of λ and x_0.*



(b) *Efficiency (2.5.5) for each combination of λ and x_0.*

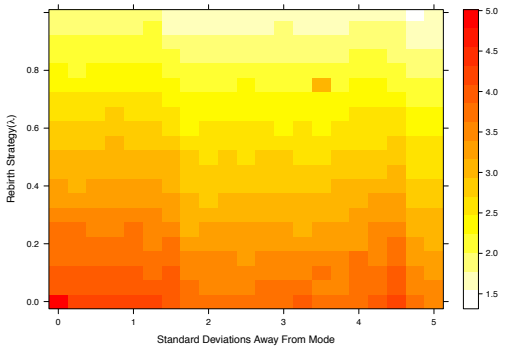


(c) *Logarithm of average computational cost in seconds.*

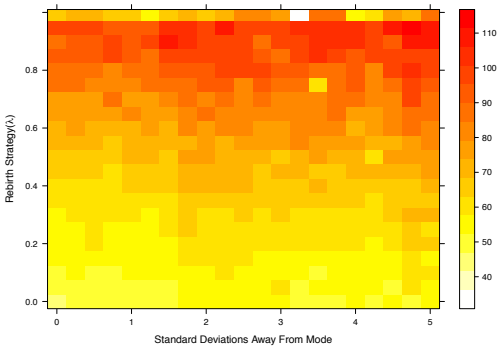


(d) *Logarithm of average uniform norm distance (2.5.2).*

Figure 7.15: *ReScaLE for Standard normal target: This experiment is based on 5 runs of the ReScaLE algorithm for a standard normal target where the maximum diffusion time $t = 10^4$.*



(a) *Computational cost (in minutes) for each combination of λ and the starting position.*



(b) *Efficiency (2.5.5) of β_0 for each combination of λ and the initial position.*

Figure 7.16: *ReScaLE for Menarche data: This experiment is based on 10 runs of the ReScaLE algorithm for a logistic regression model where the ReScaLE algorithm was run until $t = 10^6$.*

7.3 Regenerating According to a Density: Head Start Strategy

In this section, we outline another regeneration mechanism when a user has some 'prior' knowledge about the shape of the target density. Here it should be noted that we do not use a 'prior' in a Bayesian sense. A prior in our context contains the knowledge of a user about the target density under consideration. This knowledge can be then embedded within a density form, say π_0 and later used for the regeneration purpose, which will be later explained in this section. First, we lay down the motivation by illustrating a poor mixing of the ReScaLE method when the uniform regeneration strategy is invoked. We then propose the *head start* strategy where we allow our algorithm to initially regenerate according to a fixed density π_0 chosen by the user. This allow us to address the problem of a poor mixing.

In the presence of prior knowledge a user would be interested in assimilating this information during regeneration within the ReScaLE algorithm. The current implementation of our Algorithm (6.3.1) does not allow us to use such information. This can be often problematic when the target distribution is not 'well behaved', for instance when it is multi-modal. In these situations our algorithm might underexplore one of the modes. We should however note that a multi-modality is unlikely to arise in a tall data setting. We illustrate this problem through a toy example.

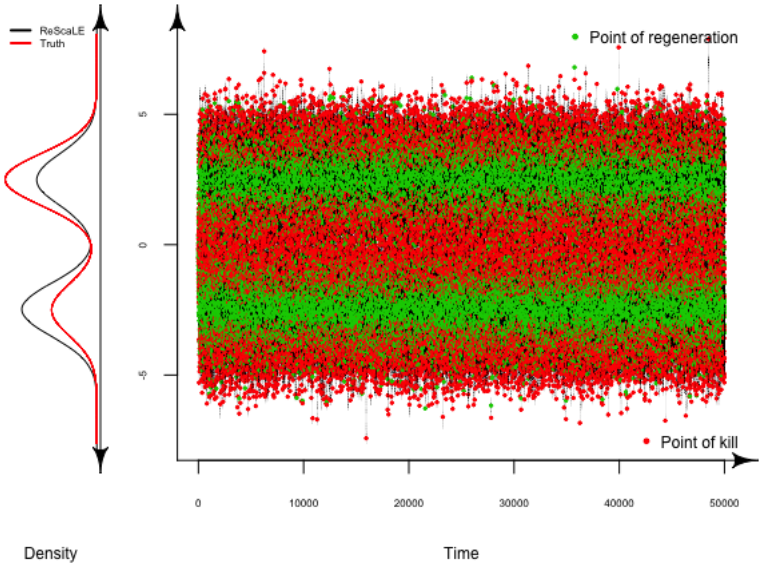
Here, we consider a target which is a mixture of two normal densities i.e.

$$\pi(x) = \frac{1}{3}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x+2.5)^2}{2}\right) + \frac{2}{3}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-2.5)^2}{2}\right). \tag{7.3.1}$$

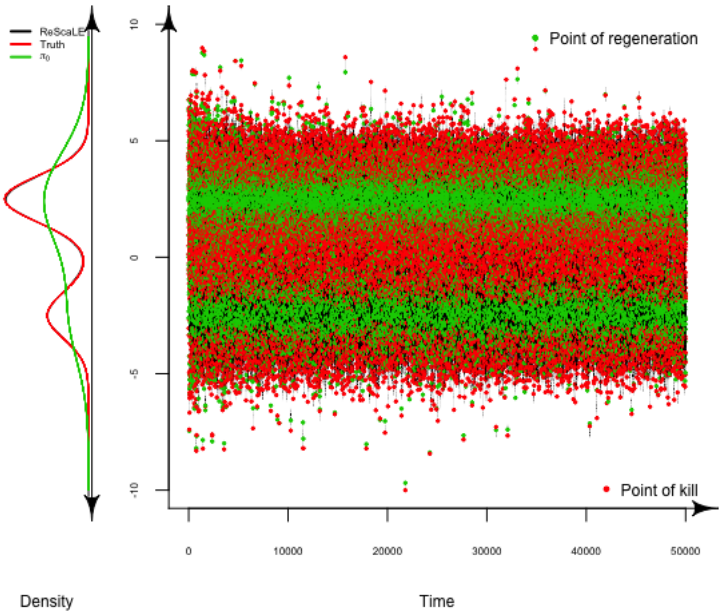
We then run our ReScaLE algorithm until $t = 5 \times 10^4$, after initialising at $x_0 = 0$. A trace-plot of the ReScaLE algorithm under the uniform regeneration strategy is presented in Figure (7.17a). From Figure (7.17a) we should note that the chain is killed instantaneously in the valley between two modes. Once it reaches one of the modes, it gets trapped within it, which explains the over-exploration of the smaller mode. Hence, there is a strong discrepancy between the kernel density estimate and the true density in Figure (7.17a) (left). The persistence within one mode can be explained as follows: once Brownian motion enters the less hazardous region (near one of the modes), it is less probable for it to travel through a valley of high hazard rates. We can observe from Figure (7.17a) that any attempt of travel through the valley gets the Brownian motion trajectory killed. Hence, our algorithm spends a disproportionate amount of time in each mode. Since ReScaLE regenerates according to its past behaviour, it is most likely to carry this disproportionate feature when subjected to a longer run. A natural question therefore arises: can the ReScaLE algorithm perform better if we allow it to regenerate according to a fixed density, which supplies partial information about the interesting regions of our target? This would provide enough information to our algorithm to learn and proportionately adjust itself within each mode.

Our motivation for regenerating according to a density can be attributed to the definition of the quasi-stationary density itself. Suppose we had access to π and we regenerate according to π whenever our trajectory is killed, the occupation measure of our trajectory will converge to π. On the contrary, π is inaccessible in our case and hence a fixed density π_0 (chosen by the user) is initially treated as a proxy for π. Therefore, one can regenerate according to π_0 when a ReScaLE trajectory is killed. However, care must be taken when π_0 is not a good representative of π. This motivates us to construct our regeneration strategy in such a way that the influence of π_0 diminishes over time. This is explained in the following paragraph.

In a more general setting, we might have some knowledge about the interesting regions of the target density. Such information can often be contained within a density, say π_0. Therefore, we can allow our algorithm to regenerate according to π_0; initially, this provides the algorithm with a head start and allows it to learn



(a)



(b)

Figure 7.17: *ReScaLE for Bimodal Example: Figure (a) illustrates the typical behaviour of the ReScaLE method under the uniform regeneration strategy. On the left, we compare the kernel density approximation against the true density. It can be observed that the ReScaLE method over-explores one of the modes. Figure (b) shows the behaviour of the algorithm under the head start strategy when the parameter* $r = 10^3$*. The prior regeneration allows the ReScaLE algorithm to explore both modes correctly; this is clear from the kernel density estimate on the left.*

about the interesting regions of the target density. However, we need to make sure that the algorithm does not over-learn according to π_0, which can result in a biased convergence. As a result we construct our *head start regeneration* strategy in such a way that the influence of our regenerating density π_0 diminishes over time and thereafter the uniform rebirth strategy starts to dominate. Following the notations of Algorithm (4.1.1) we simulate the regenerating position according to

$$\pi_t^{\mathrm{HS}} := \frac{r}{r+t} \cdot \pi_0 + \left(1 - \frac{r}{r+t}\right) \cdot \tilde{\pi}_t. \tag{7.3.2}$$

In the definition (7.3.2), $\tilde{\pi}_t$ denotes the empirical density of the states visited by the process until time t, whereas, r is a parameter supplied by the user. The tuning parameter r controls the influence of π_0 during the course of the algorithm. The parameter r also signifies the strength of a user's belief in π_0 as a proxy for the target density. The parameter r should be chosen such that the algorithm is able to gather sufficient information about the target density. The choice of the diminishing term $r/(r+t)$ is made to allow infinitely many regenerations as $t \to \infty$, which guarantees that the influence of π_0 does not decline too quickly. From definition (7.3.2) we can observe that the influence of π_0 is initially significant in the regeneration. However, as t gets large the uniform regeneration strategy of Algorithm (4.1.1) starts to dominate. Initially, this allows our algorithm to learn according to π_0 and from the empirical density in the later stages. We present the 'head start' version of Algorithm (4.1.1) in the Algorithm (7.3.1). However, we do not build the mathematical foundation for its convergence in this thesis.

Algorithm 7.3.1: SIMULATING FROM A QSD: HEAD START(π_0, r, t)

output ($\mathbf{S}$: Skeleton of a given chain)

1. Initialize the probability vector $\tilde{\pi} = \pi_0$ on the non-absorbing states $\mathcal{T}$.
2. Set $s = 0$, $\mathbf{S} \leftarrow \emptyset$ simulate $X_s \sim \tilde{\pi}$ and set $\mathbf{S} \leftarrow \mathbf{S} \cup X_s$.

while $s < t$

do
- **while** Chain is not absorbed
  - **do**
    - 3. Set $\tilde{s} = s + ds$, simulate $X_{\tilde{s}}$ according to the law of $X_{\tilde{s}} \,|\, X_s$.
    - 4. Update $\mathbf{S} \leftarrow \mathbf{S} \cup X_{\tilde{s}}$.
    - 5. Update $\tilde{\pi}$ using $\tilde{\pi}_j = \frac{1}{\tilde{s}} \int_0^{\tilde{s}} \mathbb{I}(X_q = j) dq$ for all $j \in \mathcal{T}$

    until absorption.
- 6. Simulate $U \sim U[0,1]$ and set $p = \mathbb{I}\left(U \leq \frac{r}{r+\tilde{s}}\right)$, $\pi^{\text{HS}} = p \cdot \pi_0 + (1-p) \cdot \tilde{\pi}$
- 7. Once absorbed at $\tilde{s}$, re-simulate $X_{\tilde{s}} \sim \pi^{\text{HS}}$ and set $s = \tilde{s}$.

return ($\mathbf{S}$)

We embed our head start regeneration strategy within the algorithmic construction of the ReScaLE method and use it to generate samples from the bimodal density (7.3.1). A typical trace-plot together with its kernel density estimate is presented in Figure (7.17b). In this run, the value of r was set to 10^3 and π_0 was chosen as

$$\pi_0(x) = \frac{3}{10} \frac{1}{\sqrt{2\pi \times 4}} \exp\left(-\frac{(x+2.5)^2}{2 \times 4}\right) + \frac{7}{10} \frac{1}{\sqrt{2\pi \times 4}} \exp\left(-\frac{(x-2.5)^2}{2 \times 4}\right), \tag{7.3.3}$$

which provides a vague information about the two modes (green curve in Figure (7.17b)). We then allowed the ReScaLE algorithm to run until $t = 5 \times 10^4$. We can observe that π_0 has a big influence in the initial stages of the algorithm and its effect fades away with time. This allows the ReScaLE algorithm to learn about the two modes initially and explore them proportionately in the later stages.

7.3.1 A Study with Respect to the Parameter r

Within the bimodal example (7.3.1) we perform a systematic study of the uniform norm distance with respect to the tuning parameter r. To achieve this, we obtained 5 runs of the ReScaLE algorithm for a maximum diffusion time $t = 5 \times 10^4$, each

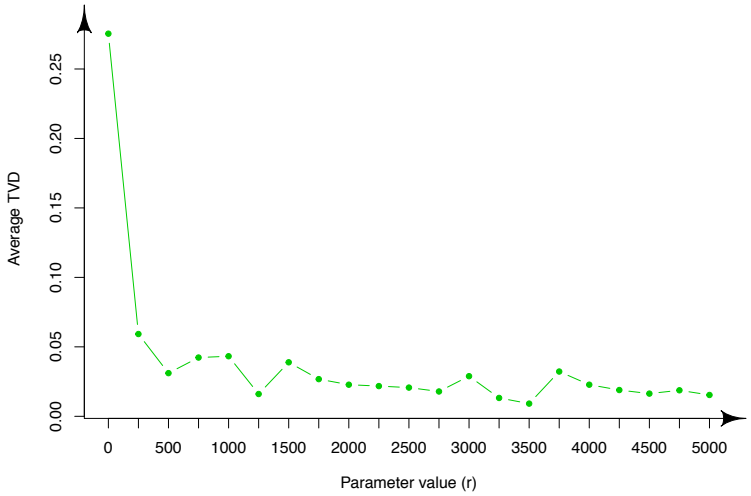


Figure 7.18: *ReScaLE for Bimodal Example: This figure illustrates the behaviour of uniform norm distance (UND,(2.5.2) between the kernel density approximation and the true density (7.3.1)) for varying values of the parameter r. In this context, a mixture of normal (7.3.3) was chosen as π_0.*

having been initialised at $x_0 = 0$. We obtain the average uniform norm distance (UND) for each parameter r, which has been demonstrated in Figure (7.18). From Figure (7.18) we can observe that the UND decreases with an increase in the value of parameter r. Because $r = 0$ corresponds to the uniform rebirth strategy, it does not allow any regeneration according to the fixed density chosen by the user. It is evident from Figure (7.18) that the ReScaLE algorithm struggles to converge to the correct target density in the absence of a head start regeneration strategy. However, as soon as we allow the influence of head start regeneration to grow (by increasing the value of r), it shows a sign of convergence (decrease in the value of UND). In this example, an elbow can be observed at a value of $r \approx 1250$ which guarantees sufficient convergence to the target density.

7.3.2 Influence of Different Head Start Densities

Next, we study the influence of different π_0 on the convergence of the ReScaLE algorithm. As noted before, the tuning parameter r controls the belief of a user in π_0 as a representative of π. When π_0 represents π very well, the tuning parameter r can be fixed to any value relative to t. On the contrary when π_0 is not a good proxy for π, setting r to a large value (relative to t) can result in a biased convergence. This is due to the fact that during a regeneration, a value for π is simulated according

to π_0. A large number of such regenerations biases our convergence towards π_0. In such situations a user should set r to a small value relative to t to ensure that the ReScaLE algorithm does not biases itself towards π_0.

We therefore consider three different π_0 : first, $U[-10, 10]$ (when a user has hardly any information about the target density and π_0 is not a good representative of π), second, we choose a mixture of normal as given in (7.3.3) (when the user has a partial information about the target and π_0 partially represents π) and finally, we use the target density itself as π_0 (when the user has a perfect information about the target density where $\pi_0 = \pi$). We then perform a systematic study of the uniform norm distance as a function of parameter r, for each π_0 considered in this experiment. To achieve this, we obtained the average UND based on 5 different runs of the ReScaLE algorithm, where each run was subjected to a maximum diffusion time, $t = 5 \times 10^4$. It can be observed from Figure (7.20) that a $U[-10, 10]$ regeneration does not provide much information to the algorithm initially, hence, the algorithm struggles to converge in a given amount of time (see for instance Figure (7.21)). However, a normal mixture (7.3.3) provides sufficient information to the ReScaLE algorithm initially. As a result our algorithm under a normal mixture regeneration converges quickly compared to a $U[-10, 10]$. We should note that allowing r to be arbitrarily large results into a biased convergence; this is evident from the divergence of the curve for large values of r. Figure (7.19) illustrates an instance when the parameter r is chosen equal to the diffusion time t. We can observe a big influence of π_0 during the course of the algorithm, which results in a biased shape of the kernel density estimate (on the left). Therefore, a user is required to make a diligent choice of π_0 and the tuning parameter r to avoid the induction of possible biases within the algorithm.

7.3.3 A Combination of Head Start with λ–rebirth Strategy

We have observed in Section (7.3.2) that a good choice of a head start density π_0 is crucial in the convergence of the ReScaLE algorithm to the desired posterior density. Often a user has very little information about the interesting regions of the posterior density and as a result, she uses a uniform density as a good candidate for π_0. We observe from Figure (7.21) that the ReScaLE algorithm is adversely influenced by the uniform distribution as a choice of π_0. This is evident from the significant number of regenerations in the tails of the posterior density which results in the discrepancy between the target and the ReScaLE-based approximation (Figure (7.21) left).

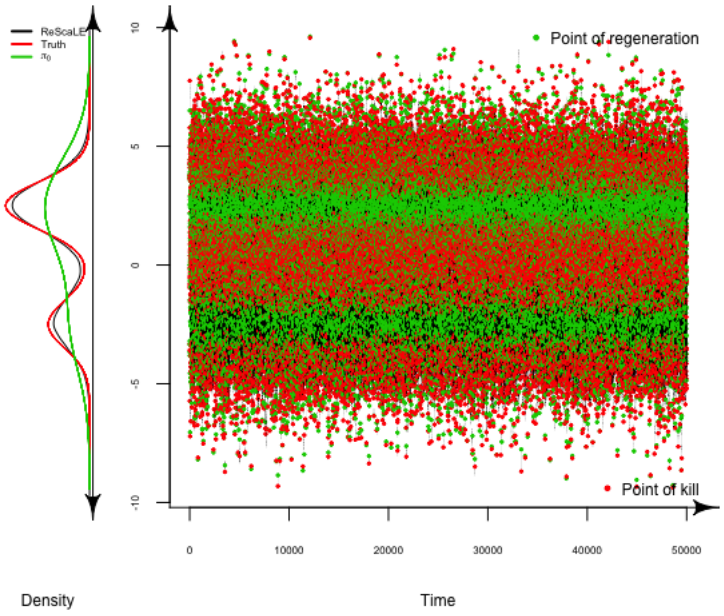


Figure 7.19: *ReScaLE for Bimodal Example: This figure illustrates the behaviour of a ReScaLE trajectory when the parameter r is chosen to be equal to the diffusion time. There is a clear discrepancy due to the excessive use of π_0 (on the left).*

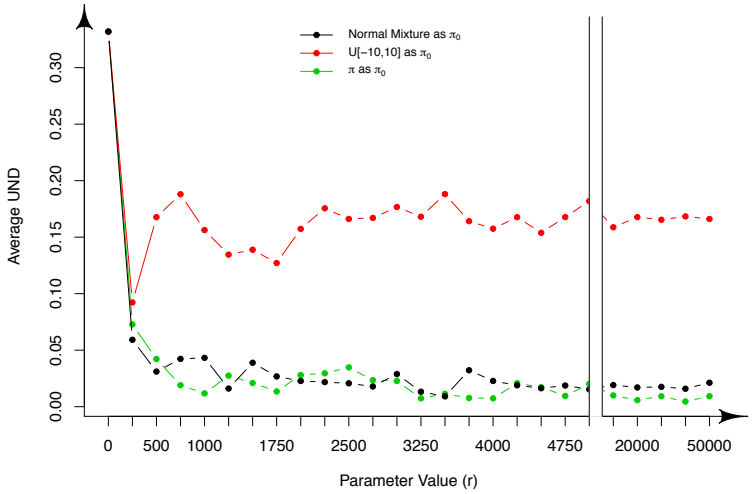


Figure 7.20: *ReScaLE for Bimodal Example: This figure illustrates the behaviour of UND (2.5.2) (between the empirical and the true distribution (7.3.1)) under the influence of different π_0, where each case is studied as a function of the tuning parameter r.*

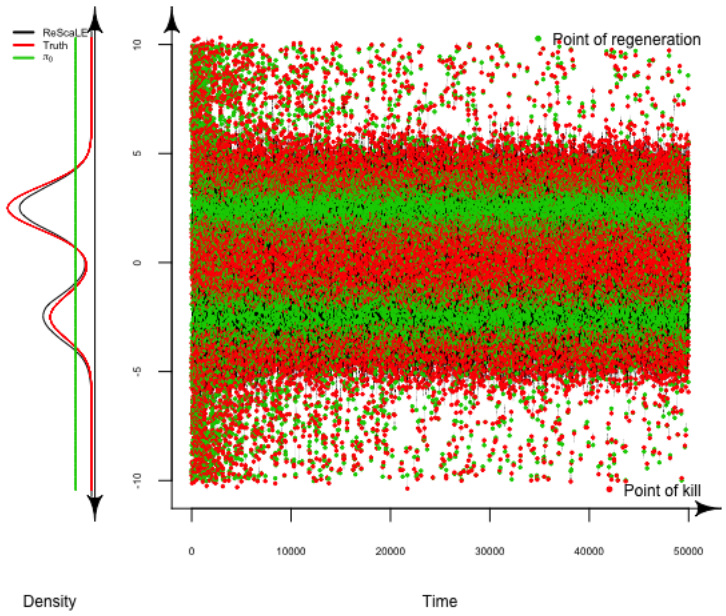


Figure 7.21: *ReScaLE for Bimodal Example: This figure illustrates a typical behaviour of the ReScaLE algorithm when a* $U[-10, 10]$ *regeneration is chosen. In this context the value of* $r = 1750$.

To circumvent this issue, we combine the head start regeneration strategy with λ–regeneration to reduce the influence of initial bias due to π_0. We have already motivated the use of the λ–regeneration strategy to reduce the effect persistence due to a poor start in Section (7.2). We experiment a combination of uniform head starts with λ–regeneration to obtain samples from (7.3.1). To illustrate this, we fix $\lambda = 0.8$, $r = 500$ and subject the ReScaLE algorithm to a maximum diffusion time of $t = 5 \times 10^5$. Figure (7.22) illustrates a typical run of the ReScaLE algorithm. In contrast to Figure (7.21) we can observe that λ–regeneration reduces the number of regenerations in the tails of the target density. Moreover, it is clear from Figure (7.21) that the convergence is achieved using a combination of uniform head starts and the λ–regeneration strategy.

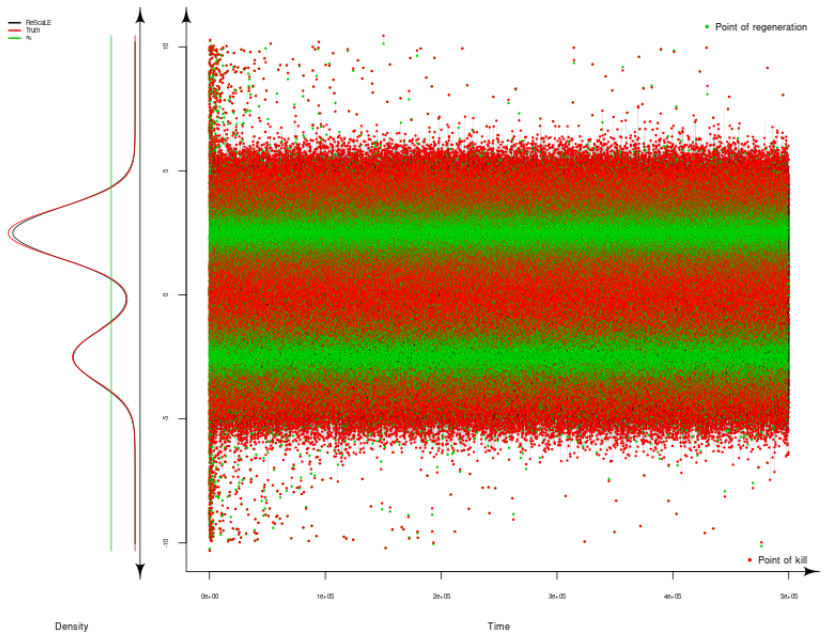


Figure 7.22: *ReScaLE for Bimodal Example: This figure illustrates a typical behaviour of the ReScaLE algorithm when a $U[-10, 10]$ head start regeneration is chosen, followed by λ–rebirth where $\lambda = 0.8$. In this context the value of $r = 500$.*

7.4 A Discussion on the Rebirth Strategies

In this chapter we have laid down the limitations of the uniform regeneration strategy of Blanchet et al. [2016] within the context of the ReScaLE algorithm. In particular, when our method was subjected to a poor start, it ran into persistence issues. A persistence problem occurs due to the strong memory of the ReScaLE algorithm under the uniform regeneration strategy of [Blanchet et al., 2016], which forces our algorithm to regenerate from a poor region. Moreover, ReScaLE experiences a poor mixing when a multi-modal target is under consideration. In this setting, our killed Brownian motion is unable to travel through regions of high hazards between the consecutive modes. As a result, our algorithm over-explores one mode and hence leads to a poor mixing.

In Section (7.2) we have witnessed a significant improvement in the performance of the ReScaLE algorithm. This improvement was achieved as a result of employing a different regeneration strategy within the ReScaLE algorithm. The empirical results illustrates that a value of λ can exist which optimizes the performance of our algorithm. However, an optimal choice of the regeneration strategy and the tuning parameter remains a challenge in a general setting. From an empirical point of view,

a user can choose a value of $\lambda \in [0.5, 0.95]$ when using a λ–rebirth strategy. We have observed that a very small choice of λ might not help the algorithm recover from the persistence issues. At the same time, a $\lambda \approx 1$ can induce biases within the ReScaLE algorithm.

In Section (7.3) we introduced the head start regeneration strategy. This is useful when a user has some prior knowledge about the shape of the target density, she can accommodate such information within a density π_0. This density can then be used to regenerate within the ReScaLE algorithm. In this context, the strength of her belief in π_0 as a proxy for π is crucial in deciding the choice of parameter r. For a fixed diffusion time t, a relatively 'small' choice of r can be made when π_0 does not represent the target density very well. However, as a big choice of r (relative to t) can result in a biased convergence, a user can alleviate this problem by running the ReScaLE algorithm for an even longer diffusion time t. We have also illustrated that a user can combine a λ–rebirth strategy within a prior regeneration when 'weak' information is provided through head start regeneration π_0.

Chapter 8

Conclusion and Further Research

In this thesis, we have introduced a QSMC method called ReScaLE which can be used as an alternative to standard MCMC algorithms, especially in a tall data setting (Chapter (6)). Contrary to the MCMC methods, ReScaLE considers a Brownian motion which possesses a killing mechanism. The hazard rate of this Brownian motion is chosen such that the intractable density of interest is given by the quasi-stationary density of the underlying process. The strength of our method lies in its scalability with respect to the data size. This was achieved by replacing the true hazard rate with its unbiased estimator, which comes as a result of employing sub-sampling together with a control variate. This offers an unbiased calculation of the hazard rate at an expense of cheaper computational cost, however, it still guarantees us the correct target density [Pollock et al., 2017].

We have empirically shown that in light of sub-sampling with the control variate information, the method possesses a sub-linear scalability with respect to the data size. Although the computational cost involved in the calculation of the control variate information is $\mathcal{O}(n)$, this needs to be calculated only once; at the start of the ReScaLE algorithm. We have empirically validated that a standard MCMC algorithm becomes highly inefficient in a big data setting. The ReScaLE method however, retains its efficiency with respect to an increasing data size. Furthermore, we illustrated the working of the ReScaLE method on a real-world tall data problem, where we considered the US domestic airline dataset. The aim was to sample according to the posterior distribution of a logistic regression model which predicts whether or not a given flight is delayed. This example shows the applicability of the

ReScaLE algorithm in a big data setting where a standard MCMC method can be highly inefficient.

As opposed to the existing QSMC method ScaLE, the ReScaLE method provides a reduction in the computational cost involved in simulating according to the quasi-stationary density of a killed Brownian motion. While the ScaLE method employs an SMC-based approach to target the quasi-stationary density, ReScaLE utilizes a natural extension of the regenerative approach of Blanchet et al. [2016]. The simplicity and use of a single trajectory in the regenerative approach allows us to gain the computational advantage over the ScaLE method.

This thesis further proposes other regeneration techniques to address the problem of a poor start within the ReScaLE algorithm. We have noted that within the ReScaLE algorithm, the regenerative mechanism of Blanchet et al. [2016] suffers from persistence issues if it is subjected to a poor start. This situation is of particular importance when the user does not have sufficient information about the posterior mode. Therefore, an arbitrarily initialised ReScaLE algorithm is likely to run into a persistence problem. To address the problem of a poor start, we proposed a λ–rebirth strategy where samples were regenerated from the 'recent' past. The idea was to allow our algorithm to forget the earlier part of the trajectory which over-represents the unlikely part of the state-space. We have empirically illustrated that a problem of a poor start can be addressed by employing a λ–rebirth strategy within the algorithmic construction of the ReScaLE method. The empirical findings show that while an optimal choice of λ can be made, an optimal value of λ can vary from one problem to the another.

We further proposed a head start regeneration strategy where a user can regenerate according to a density of her choice, whose influence diminishes with an increasing value of diffusion time. A head start regeneration is useful when a user has some knowledge about the target density which she can embed within a density (π_0) and can be used to regenerate in the initial phase of the algorithm. We have empirically validated the working of a head start regeneration strategy in a bimodal example, where the ReScaLE algorithm possesses a poor mixing property under the uniform regeneration. A user however needs to make a diligent choice of π_0 and the parameter r to ensure convergence to the correct target density.

In spite of the empirical strengths shown in this thesis, the ReScaLE method still

faces some key challenges:

- the theoretical justification of the regenerative algorithm by Blanchet et al. 2016 is yet to be guaranteed over a general state-space. On the positive side, Wang et al. 2017 has proved its validity over a compact state-space setting.

- A super-efficient calculation of the control variate information which can beat its usual $\mathcal{O}(n)$ computational complexity, has not been addressed. The calculation of control variates can often be challenging, especially when a large dataset is under consideration. At the same time, control variates are critical in the implementation of the ReScaLE method; an incorrectly observed control variate information might force the algorithm to explore the 'bad' regions of the target density. Furthermore, when the posterior density possesses multiple modes, a single control variate information can be often 'misleading' which can force the algorithm into persistence issues. Therefore, the current implementation of the ReScaLE algorithm requires a framework which can employ multiple control variate information at the same time.

- The current implementation of the ReScaLE method faces persistence issues in the multi-modal setting, especially when the modes are separated far apart. This is mainly due to the fact that the underlying killed Brownian motion is unable to travel through the region of high hazard between the modes of the target density. The currently proposed rebirth strategies do not work in these settings. This is still theoretically unexplored and hence it requires consideration in future research.

- Although the empirical findings for the proposed rebirth strategies are promising, their mathematical validity has not been studied in this thesis. This does not only open an interesting avenue of research, but also looks at questions about the existence of an optimal rebirth strategy which works in a generic setting.

- The current implementation of existing QSMC methods (namely ScaLE and ReScaLE) lack the theoretical support which justifies their algorithmic complexity with respect to the dimensionality of a given problem. Similar to sub-sampling over data points, sub-sampling over dimensions would be an interesting avenue for further research, which may in turn reduce its computational complexity.

The solutions to the problems mentioned above would not only offer the required theoretical support to the ReScaLE method but at the same time, circumvent the

current limitations faced by the method. Unlike MCMC, the field of QSMC is fairly recent of which ScaLE and ReScaLE are only two existing algorithms. Therefore, we conclude by stating that new approaches to the quasi-stationary simulations would enrich the field of QSMC, which can then be blended within the QSMC theory to design more efficient algorithms to sample from an intractable distribution of interest.

Part III

Bibliography

Bibliography

H. Andersson and T. Britton. Stochastic Epidemics in Dynamic Populations: Quasi-stationarity and Extinction. *J. Math. Biol.*, 41(6):559–580, 2000. ISSN 0303-6812. doi: 10.1007/s002850000060. URL https://doi.org/10.1007/s002850000060.

F. J. Aranda-Ordaz. On Two Families of Transformations to Additivity for Binary Response Data. *Biometrika*, 68(2):357–363, 1981. ISSN 0006-3444. doi: 10.1093/biomet/68.2.357. URL https://doi.org/10.1093/biomet/68.2.357.

E. Arnaud and E. Mémin. Partial Linear Gaussian Models for Tracking in Image Sequences using Sequential Monte Carlo Methods. *Int J Comput Vis*, 74(1):75–102, 2007.

A. Asselah, P. A. Ferrari, and P. Groisman. Quasistationary Distributions and Fleming-Viot Processes in Finite Spaces. *J. Appl. Probab.*, 48(2):322–332, 2011. ISSN 0021-9002. doi: 10.1239/jap/1308662630. URL https://doi.org/10.1239/jap/1308662630.

J. Baker, P. Fearnhead, E. B. Fox, and C. Nemeth. Control Variates for Stochastic Gradient MCMC. *arXiv preprint arXiv:1706.05439*, 2017.

R. Bardenet, A. Doucet, and C. Holmes. On Markov Chain Monte Carlo Methods for Tall Data. *J. Mach. Learn. Res.*, 18:Paper No. 47, 43, 2017. ISSN 1532-4435.

T. Bengtsson, P. Bickel, and B. Li. Curse-of-dimensionality Revisited: Collapse of the Particle Filter in Very Large Scale Systems. In *Probability and Statistics: Essays in Honor of David A. Freedman*, volume 2 of *Inst. Math. Stat. (IMS) Collect.*, pages 316–334. Inst. Math. Statist., Beachwood, OH, 2008. doi: 10.1214/193940307000000518. URL https://doi.org/10.1214/193940307000000518.

A. Beskos and G. O. Roberts. Exact Simulation of Diffusions. *Ann Appl Probab*, 15 (4):2422–2444, 2005. ISSN 10505164. doi: 10.1214/105051605000000485.

A. Beskos, O. Papaspiliopoulos, and G. O. Roberts. Retrospective Exact Simulation of Diffusion Sample Paths with Applications. *Bernoulli*, 12(6):1077–1098, 2006. ISSN 1350-7265. doi: 10.3150/bj/1165269151. URL https://doi.org/10.3150/bj/1165269151.

A. Beskos, O. Papaspiliopoulos, and G. O. Roberts. A Factorisation of Diffusion Measure and Finite Sample Path Constructions. *Methodol. Comput. Appl. Probab.*, 10(1):85–104, 2008. ISSN 1387-5841. doi: 10.1007/s11009-007-9060-4. URL https://doi.org/10.1007/s11009-007-9060-4.

M. Bieniek, K. Burdzy, and S. Pal. Extinction of Fleming-Viot-type Particle Systems with Strong Drift. *Electron. J. Probab.*, 17:no. 11, 15, 2012. ISSN 1083-6489. doi: 10.1214/EJP.v17-1770. URL https://doi.org/10.1214/EJP.v17-1770.

J. Bierkens, P. Fearnhead, and G. O. Roberts. The Zig-Zag Process and Super-efficient Sampling for Bayesian Analysis of Big Data. *arXiv preprint arXiv:1607.03188*, 2016.

J. Blanchet, P. Glynn, and S. Zheng. Analysis of a Stochastic Approximation Algorithm for Computing Quasi-stationary Distributions. *Adv. in Appl. Probab.*, 48(3):792–811, 2016. ISSN 0001-8678. doi: 10.1017/apr.2016.28. URL https://doi.org/10.1017/apr.2016.28.

A. Bouchard-Côté, S. J. Vollmer, and A. Doucet. The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method. *J Am Stat Assoc*, 113(522):855–867, 2018. doi: 10.1080/01621459.2017.1294075. URL https://doi.org/10.1080/01621459.2017.1294075.

N. Bouleau and D. Lépingle. *Numerical Methods for Stochastic Processes*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons, Inc., New York, 1994. ISBN 0-471-54641-0. A Wiley-Interscience Publication.

P. Brasnett, L. Mihaylova, D. Bull, and N. Canagarajah. Sequential Monte Carlo Tracking by Fusing Multiple Cues in Video Sequences. *Image Vis Comput*, 25(8): 1217–1227, 2007.

S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, editors. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, Boca Raton, FL, 2011. ISBN 978-1-4200-7941-8. doi: 10.1201/b10905. URL https://doi.org/10.1201/b10905.

S. P. Brooks and A. Gelman. General Methods for Monitoring Convergence of Iterative Simulations. *J. Comput. Graph. Statist.*, 7(4):434–455, 1998. ISSN 1061-8600. doi: 10.2307/1390675. URL https://doi.org/10.2307/1390675.

K. Burdzy, R. Holyst, D. Ingerman, and P. March. Configurational Transition in a Fleming-Viot-type Model and Probabilistic Interpretation of Laplacian Eigenfunctions. *J. Phys. A: Math. Gen.*, 29(11):2633, 1996.

K. Burdzy, R. Holyst, and P. March. A Fleming-Viot Particle Representation of the Dirichlet Laplacian. *Comm. Math. Phys.*, 214(3):679–703, 2000. ISSN 0010-3616. doi: 10.1007/s002200000294. URL https://doi.org/10.1007/s002200000294.

Z. A. Burq and O. D. Jones. Simulation of Brownian Motion at First-passage Times. *Math. Comput. Simulation*, 77(1):64–71, 2008. ISSN 0378-4754. doi: 10.1016/j.matcom.2007.01.038. URL https://doi.org/10.1016/j.matcom.2007.01.038.

N. Chen and Z. Huang. Localization and Exact Simulation of Brownian Motion-driven Stochastic Differential Equations. *Math. Oper. Res.*, 38(3):591–616, 2013. ISSN 0364-765X. doi: 10.1287/moor.2013.0585. URL https://doi.org/10.1287/moor.2013.0585.

P. Collet, S. Martínez, and J. San Martín. *Quasi-stationary Distributions: Markov Chains, Diffusions and Dynamical Systems*. Springer Science & Business Media, 2012.

D. R. Cox and V. Isham. *Point Processes*. Chapman & Hall, London-New York, 1980. ISBN 0-412-21910-7. Monographs on Applied Probability and Statistics.

D. Dacunha-Castelle and D. Florens-Zmirou. Estimation of the Coefficients of a Diffusion from Discrete Observations. *Stochastics*, 19(4):263–284, 1986. ISSN 0090-9491. doi: 10.1080/17442508608833428. URL https://doi.org/10.1080/17442508608833428.

S. Dambrine and M. Moreau. Note on the Stochastic Theory of a Self-catalytic Chemical Reaction. I. *Phys. A*, 106(3):559–573, 1981a.

S. Dambrine and M. Moreau. Note on the Stochastic Theory of a Self-catalytic Chemical Reaction. I, II. *Phys. A*, 106(3):559–588, 1981b. ISSN 0378-4371. doi: 10.1016/0378-4371(81)90126-6. URL https://doi.org/10.1016/0378-4371(81)90126-6.

J. N. Darroch and E. Seneta. On Quasi-stationary Distributions in Absorbing Discrete-time Finite Markov Chains. *J. Appl. Probability*, 2:88–100, 1965. ISSN 0021-9002. doi: 10.2307/3211876. URL https://doi.org/10.2307/3211876.

J. N. Darroch and E. Seneta. On Quasi-stationary Distributions in Absorbing Continuous-time Finite Markov Chains. *J. Appl. Probability*, 4:192–196, 1967. ISSN 0021-9002. doi: 10.2307/3212311. URL https://doi.org/10.2307/3212311.

M. de Oliveira and R. Dickamn. How to Simulate the Quasi-stationary State. *Phys. Rev. E*, 71(1):9, 2005. doi: 10.1103/PhysRevE.71.016129. URL http://link.aps.org/doi/10.1103/PhysRevE.71.016129.

P. de Valpine, D. Turek, C. J. Paciorek, C. Anderson-Bergman, D. Temple Lang, and R. Bodik. Programming with Models: Writing Statistical Algorithms for General Model Structures with NIMBLE. *J. Comput. Graph. Statist.*, 26(2): 403–413, 2017. ISSN 1061-8600. doi: 10.1080/10618600.2016.1172487. URL https://doi.org/10.1080/10618600.2016.1172487.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo Samplers. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 68(3):411–436, 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2006.00553.x. URL https://doi.org/10.1111/j.1467-9868.2006.00553.x.

P. Del Moral, A. Doucet, and A. Jasra. On Adaptive Resampling Procedures for Sequential Monte Carlo Methods. HAL-INRIA RR-6700-2008. *Bernoulli*, pages 2496–2534, 2011.

L. Devroye. Generating the Maximum of Independent Identically Distributed Random Variables. *Comput. Math. Appl.*, 6(3):305–315, 1980. ISSN 0097-4943. doi: 10.1016/0898-1221(80)90039-5. URL https://doi.org/10.1016/0898-1221(80)90039-5.

L. Devroye. *Non-Uniform Random Variate Generation*. Springer New York, 1986. ISBN 9783540963059. URL https://books.google.co.uk/books?id=mEw_AQAAIAAJ.

A. Doucet and A. M. Johansen. A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later. In *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford Univ. Press, Oxford, 2011.

A. Doucet, S. Godsill, and C. Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Stat Comput*, 10(3):197–208, 2000.

J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011. ISSN 1532-4435.

D. Eddelbuettel. *Seamless R and C++ integration with Rcpp*. Springer, 2013.

D. Eddelbuettel. CRAN Task View: High-performance and Parallel Computing with R. 2018.

V. c. Elvira, J. n. Mí guez, and P. M. Djurić. Adapting the Number of Particles in Sequential Monte Carlo Sethods through an Online Scheme for Convergence Assessment. *IEEE Trans. Signal Process.*, 65(7):1781–1794, 2017. ISSN 1053-587X. doi: 10.1109/TSP.2016.2637324. URL https://doi.org/10.1109/TSP.2016.2637324.

W. J. Ewens. The Diffusion Equation and a Pseudo-distribution in Genetics. *J. Roy. Statist. Soc. Ser. B*, 25:405–412, 1963. ISSN 0035-9246. URL http://links.jstor.org/sici?sici=0035-9246(1963)25:2<405:TDEAAP>2.0.CO;2-Z&origin=MSN.

W. J. Ewens. The Pseudo-transient Distribution and its Uses in Genetics. *J. Appl. Probability*, 1:141–156, 1964. ISSN 0021-9002. doi: 10.2307/3212065. URL https://doi.org/10.2307/3212065.

P. A. Ferrari and N. Marić. Quasi Stationary Distributions and Fleming-Viot Processes in Countable Spaces. *Electron. J. Probab.*, 12:no. 24, 684–702, 2007. ISSN 1083-6489. doi: 10.1214/EJP.v12-415. URL https://doi.org/10.1214/EJP.v12-415.

P. A. Ferrari, S. Martinez, and P. Picco. Some Properties of Quasi-stationary Distributions in the Birth and Death Chains: a Dynamical Approach. In *Instabilities and Nonequilibrium Structures, III (Valparaíso, 1989)*, volume 64 of *Math. Appl.*, pages 177–187. Kluwer Acad. Publ., Dordrecht, 1991.

J. M. Flegal, J. Hughes, D. Vats, and N. Dai. *mcmcse: Monte Carlo Standard Errors for MCMC*. Riverside, CA, Denver, CO, Coventry, UK, and Minneapolis, MN, 2017. R package version 1.3-2.

W. H. Fleming and M. Viot. Some Measure-valued Markov Processes in Population Genetics Theory. *Indiana Univ. Math. J.*, 28(5):817–843, 1979. ISSN 0022-2518. doi: 10.1512/iumj.1979.28.28058. URL https://doi.org/10.1512/iumj.1979.28.28058.

W. Fong, S. J. Godsill, A. Doucet, and M. West. Monte Carlo Smoothing with Application to Audio Signal Enhancement. *IEEE Trans. Signal Process.*, 50(2): 438–449, 2002.

O. Frank, J. Nieto, J. Guivant, and S. Scheding. Multiple Target Tracking using Sequential Monte Carlo Methods and Statistical Data Association. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2718–2723. IEEE, 2003.

A. Gelman and D. B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statist. Sci.*, 7(4):457–472, 1992. ISSN 08834237. URL http://www.jstor.org/stable/2246093.

A. Gelman and K. Shirley. Inference from Simulations and Monitoring Convergence. In *Handbook of Markov Chain Monte Carlo*, Chapman & Hall/CRC Handb. Mod. Stat. Methods, pages 163–174. CRC Press, Boca Raton, FL, 2011. doi: 10.1201/b10905. URL https://doi.org/10.1201/b10905.

A. Gelman, A. Jakulin, M. G. Pittau, and Y.-S. Su. A weakly informative default prior distribution for logistic and other regression models. *Ann. Appl. Stat.*, 2 (4):1360–1383, 12 2008. doi: 10.1214/08-AOAS191. URL https://doi.org/10.1214/08-AOAS191.

A. Gelman, S. Brooks, G. Jones, and X.-L. Meng. Introduction to Markov Chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo*, pages 24–70. Chapman and Hall/CRC, 2011.

J. Geweke. Bayesian Inference in Econometric Models using Monte Carlo Integration. *Econometrica*, 57(6):1317–1339, 1989. ISSN 0012-9682. doi: 10.2307/1913710. URL https://doi.org/10.2307/1913710.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Interdisciplinary Statistics. Chapman & Hall, London, 1996. ISBN 0-412-05551-1. doi: 10.1007/978-1-4899-4485-6. URL https://doi.org/10.1007/978-1-4899-4485-6.

S. Godsill, P. Rayner, and O. Cappé. Digital Audio Restoration. In *Applications of Digital Signal Processing to Audio and Acoustics*, pages 133–194. Springer, 2002.

G. Goertzel, U. A. E. Commission, and O. R. N. Laboratory. *Quota Sampling and Importance Functions in Stochastic Solution of Particle Problems*. AECD. U.S. Atomic Energy Commission, Technical Information Division, 1950. URL https://books.google.co.uk/books?id=Su1EGYGagoAC.

N. J. Gordon, D. J. Salmond, and A. F. Smith. Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

I. Grigorescu and M. Kang. Hydrodynamic Limit for a Fleming-Viot–type System. *Stochastic Process. Appl.*, 110(1):111–143, 2004. ISSN 0304-4149. doi: 10.1016/j.spa.2003.10.010. URL https://doi.org/10.1016/j.spa.2003.10.010.

P. Groisman and M. Jonckheere. Simulation of Quasi-stationary Distributions on Countable Spaces. *Markov Process. Related Fields*, 19(3):521–542, 2013. ISSN 1024-2953.

D. Gunawan, M.-N. Tran, and R. Kohn. Fast Inference for Intractable Likelihood Problems using Variational Bayes. *arXiv preprint arXiv:1705.06679*, 2017.

Z. Huang and A. Gelman. Sampling for Bayesian Computation with Large Datasets. 2005. URL https://dx.doi.org/10.2139/ssrn.1010107.

C. Hue, J.-P. Le Cadre, and P. Perez. Sequential Monte Carlo Methods for Multiple Target Tracking and Data Fusion. *IEEE Trans. Signal Process*, 50(2):309–325, 2002.

H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big Data and Its Technical Challenges. *Commun. ACM*, 57(7):86–94, July 2014. ISSN 0001-0782. doi: 10.1145/2611567. URL http://0-doi.acm.org.pugwash.lib.warwick.ac.uk/10.1145/2611567.

H. Kahn. Stochastic (Monte Carlo) Attenuation Analysis. Technical report, Rand Corp Santa Monica Calif, 1949.

M. J. Kane. *Scalable Strategies for Computing with Massive Sets of Data*. ProQuest LLC, Ann Arbor, MI, 2010. ISBN 978-1124-42282-4. URL http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:3440560. Thesis (Ph.D.)–Yale University.

I. Karatzas and S. E. Shreve. *Brownian Motion and Stochastic Calculus*, volume 113. Springer, New York, 2nd edition, 1991. ISBN 0387976558. doi: 10.1007/978-1-4612-0949-2. URL http://www.amazon.com/dp/0387976558.

S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes*. Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, second edition, 1975.

G. E. Karniadakis and R. M. Kirby II. *Parallel Scientific Computing in C++ and MPI: a Seamless Approach to Parallel Algorithms and their Implementation*. Cambridge University Press, 2003.

A. Katal, M. Wazid, and R. Goudar. Big Data: Issues, Challenges, Tools and Good Practices. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 404–409. IEEE, 2013.

F. P. Kelly. Stochastic Models of Computer Communication Systems. *J. Roy. Statist. Soc. Ser. B*, 47(3):379–395, 415–428, 1985. ISSN 0035-9246. URL http://links.jstor.org/sici?sici=0035-9246(1985)47:3<379:SMOCCS>2.0.CO;2-F&origin=MSN. With discussion.

J. F. C. Kingman. *Poisson Processes*, volume 3 of *Oxford Studies in Probability*. The Clarendon Press, Oxford University Press, New York, 1993. ISBN 0-19-853693-3. Oxford Science Publications.

P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, Berlin, 3rd edition, 1999.

H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*, volume 35 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, second edition, 2003. ISBN 0-387-00894-2. Stochastic Modelling and Applied Probability.

A. Labrinidis and H. V. Jagadish. Challenges and Opportunities with Big Data. *Proceedings of the VLDB Endowment*, 5(12):2032–2033, 2012.

J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. Springer, New York, 2008. ISBN 978-0-387-76369-9; 0-387-95230-6.

D. Maclaurin and R. P. Adams. Firefly Monte Carlo: Exact MCMC with Subsets of Data. In *UAI*, pages 543–552, 2014.

C. Mailler and D. Villemonais. Stochastic Approximation on Non-compact Measure Spaces and Application to Measure-valued Pólya Processes. sep 2018. URL http://arxiv.org/abs/1809.01461.

R. Mansuy and M. Yor. *Aspects of Brownian Motion*. Universitext. Springer-Verlag, Berlin, 2008. ISBN 978-3-540-22347-4. doi: 10.1007/978-3-540-49966-4. URL https://doi.org/10.1007/978-3-540-49966-4.

A. D. Martin, K. M. Quinn, and J. H. Park. MCMCpack: Markov Chain Monte Carlo in R. *J Stat Softw*, 42(9):22, 2011. URL http://www.jstatsoft.org/v42/i09/.

N. Matloff. *Parallel Computing for Data Science: with Examples in R, C++ and CUDA*. Chapman and Hall/CRC, 2015.

S. Méléard, D. Villemonais, et al. Quasi-stationary Distributions and Population Processes. *Probab. Surv.*, 9:340–410, 2012.

I. Nå sell. On the Time to Extinction in Recurrent Epidemics. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 61(2):309–330, 1999. ISSN 1369-7412. doi: 10.1111/1467-9868.00178. URL https://doi.org/10.1111/1467-9868.00178.

T. Nagapetyan, A. B. Duncan, L. Hasenclever, S. J. Vollmer, L. Szpruch, and K. Zygalakis. The True Cost of Stochastic Gradient Langevin Dynamics. *arXiv preprint arXiv:1706.02692*, 2017.

W. Neiswanger, C. Wang, and E. P. Xing. Asymptotically Exact, Embarrassingly Parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pages 623–632, Arlington, Virginia, United States, 2013. AUAI Press. ISBN 978-0-9749039-1-0. URL http://dl.acm.org/citation.cfm?id=3020751.3020816.

W. Oçafrain and D. Villemonais. Convergence of a Non-failable Mean-field Particle System. *Stoch. Anal. Appl.*, 35(4):587–603, 2017. ISSN 0736-2994. doi: 10.1080/07362994.2017.1288136. URL https://doi.org/10.1080/07362994.2017.1288136.

B. Øksendal. *Stochastic Differential Equations*. Springer-Verlag Berlin Heidelberg, Berlin, 6th edition, 2003. ISBN 978-3-540-04758-2. doi: 10.1007/978-3-642-14394-6. URL http://dx.doi.org/10.1007/978-3-642-14394-6{_}5.

P. D. O'Neill. Constructing Population Processes with Specified Quasi-stationary Distributions. *Stoch. Models*, 23(3):439–449, 2007. ISSN 1532-6349. doi: 10.1080/15326340701471075. URL https://doi.org/10.1080/15326340701471075.

J. Owen, D. J. Wilkinson, and C. S. Gillespie. Scalable Inference for Markov Processes with Intractable Likelihoods. *Stat Comput*, 25(1):145–156, nov 2014. doi: 10.1007/s11222-014-9524-7. URL https://doi.org/10.1007%2Fs11222-014-9524-7.

A. G. Pakes. Limit Theorems for the Population Size of a Birth and Death Process Allowing Catastrophes. *J. Math. Biol.*, 25(3):307–325, 1987. ISSN 0303-6812. doi: 10.1007/BF00276439. URL https://doi.org/10.1007/BF00276439.

A. G. Pakes and P. K. Pollett. The Supercritical Birth, Death and Catastrophe Process: Limit Theorems on the set of Extinction. *Stochastic Process. Appl.*, 32 (1):161–170, 1989. ISSN 0304-4149. doi: 10.1016/0304-4149(89)90060-4. URL https://doi.org/10.1016/0304-4149(89)90060-4.

M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence Diagnosis and Output Analysis for MCMC. *R News*, 6(1):7–11, 2006. URL https://journal.r-project.org/archive/.

P. K. Pollett. Modelling Random Fluctuations in a Bistable Telecommunications Network. In *Proceedings of the 11th National Conference of the Australian Society for Operations Research*, pages 11–22.

M. Pollock. *Some Monte Carlo Methods for Jump Diffusions*. ProQuest LLC, Ann Arbor, MI, 2013. URL http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqm&rft_dat=xri:pqdiss:U636634. Thesis (Ph.D.)–University of Warwick (United Kingdom).

M. Pollock, P. Fearnhead, A. M. Johansen, and G. O. Roberts. The Scalable Langevin Exact Algorithm: Bayesian Inference for Big Data. *ArXiv e-prints*, Sept. 2017.

K. Pötzelberger and L. Wang. Boundary Crossing Probability for Brownian Motion. *J. Appl. Probab.*, 38(1):152–164, 2001. ISSN 0021-9002. doi: 10.1017/s002190020001857x. URL https://doi.org/10.1017/s002190020001857x.

M. B. Priestley. *Spectral Analysis and Time Series. Vol. 1.* Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], London-New York, 1981. ISBN 0-12-564901-0. Univariate series, Probability and Mathematical Statistics.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

A. Racine, A. P. Grieve, H. Fluhler, and A. F. M. Smith. Bayesian methods in practice: Experiences in the pharmaceutical industry. *J R Stat Soc Ser C Appl Stat*, 35(2):93–150, 1986. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2347264.

A. E. Raftery and S. M. Lewis. The Number of Iterations, Convergence Diagnostics and Generic Metropolis Algorithms. In *In Practical Markov Chain Monte Carlo (W.R. Gilks, D.J. Spiegelhalter and*, pages 115–130. Chapman and Hall, 1995.

D. Revuz and M. Yor. *Continuous Martingales and Brownian Motion*, volume 293 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, third edition, 1999. ISBN 3-540-64325-7. doi: 10.1007/978-3-662-06400-9. URL https://doi.org/10.1007/978-3-662-06400-9.

H. Robbins and S. Monro. A Stochastic Approximation Method. *Ann. Math. Statistics*, 22:400–407, 1951. ISSN 0003-4851. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Science & Business Media, 2013.

S. Ross. *Introduction to Probability Models*. Academic Press, Inc., Orlando, FL, USA, 10$^{\text{th}}$ edition, 2010. ISBN 9780123756862. doi: 10.1080/00401706.1998.10485493.

S. Ross. *Simulation*. Elsevier Science, 2012. ISBN 9780124158252. URL https://books.google.co.uk/books?id=1Dwsyyty3P8C.

I. Sato and H. Nakagawa. Approximation Analysis of Stochastic Gradient Langevin Dynamics by using Fokker-Planck Equation and Ito Process . In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 982–990,

Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/satoa14.html.

S. L. Scott. Comparing Consensus Monte Carlo Strategies for Distributed Bayesian Computation. *Braz. J. Probab. Stat.*, 31(4):668–685, 2017. ISSN 0103-0752. doi: 10.1214/17-BJPS365. URL https://doi.org/10.1214/17-BJPS365.

S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch. Bayes and Big Data: the Consensus Monte Carlo Algorithm. *International Journal of Management Science and Engineering Management*, 11 (2):78–88, 2016. doi: 10.1080/17509653.2016.1142191. URL https://doi.org/10.1080/17509653.2016.1142191.

E. Seneta and D. Vere-Jones. On Quasi-stationary Distributions in Discrete-time Markov Chains with a Denumerable Infinity of States. *J. Appl. Probability*, 3: 403–434, 1966. ISSN 0021-9002. doi: 10.2307/3212128. URL https://doi.org/10.2307/3212128.

C. Sherlock, P. Fearnhead, and G. O. Roberts. The Random Walk Metropolis: Linking Theory and Practice Through a Case Study. *Statist. Sci.*, 25(2):172–190, 2010. ISSN 0883-4237. doi: 10.1214/10-STS327. URL https://doi.org/10.1214/10-STS327.

A. W. van der Vaart. *Asymptotic Statistics*, volume 3 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 1998. ISBN 0-521-49603-9; 0-521-78450-6. doi: 10.1017/CBO9780511802256. URL https://doi.org/10.1017/CBO9780511802256.

E. A. van Doorn. Quasi-stationary Distributions and Convergence to Quasi-stationarity of Birth-death Processes. *Adv. in Appl. Probab.*, 23(4):683–700, 1991. ISSN 0001-8678. doi: 10.2307/1427670. URL https://doi.org/10.2307/1427670.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-Plus*. Statistics and Computing. Springer-Verlag, New York, 1994. ISBN 0-387-94350-1. doi: 10.1007/978-1-4899-2819-1. URL https://doi.org/10.1007/978-1-4899-2819-1. With 1 IBM-PC floppy disk (3.5 inch; HD).

J. Vermaak, C. Andrieu, A. Doucet, and S. J. Godsill. Particle Methods for Bayesian Modeling and Enhancement of Speech Signals. *IEEE Trans. Speech Audio Process.*, 10(3):173–185, 2002.

D. Villemonais. Interacting Particle Systems and Yaglom Limit Approximation of Diffusions with Unbounded Drift. *Electron. J. Probab.*, 16:no. 61, 1663–1692, 2011a. ISSN 1083-6489. doi: 10.1214/EJP.v16-925. URL https://doi.org/10.1214/EJP.v16-925.

D. Villemonais. Interacting Particle Processes and Approximation of Markov Processes Conditioned to not be Killed. *ArXiv e-prints*, 2011b.

J. von Neumann. Various Techniques Used in Connection with Random Digits. In A. Householder, G. Forsythe, and H. Germond, editors, *Monte Carlo Method*, pages 36–38. National Bureau of Standards Applied Mathematics Series, 12, Washington, D.C.: U.S. Government Printing Office, 1951.

A. Q. Wang, M. Kolb, G. O. Roberts, and D. Steinsaltz. Theoretical Properties of Quasi-stationary Monte Carlo Methods. *arXiv preprint*, jul 2017. URL http://arxiv.org/abs/1707.08036. Accepted for publication in Ann. Appl. Probab.

A. Q. Wang, G. O. Roberts, and D. Steinsaltz. An Approximation Scheme for Quasi-stationary Distributions of Killed Diffusions. aug 2018. URL http://arxiv.org/abs/1808.07086.

C. Wang, M.-H. Chen, E. Schifano, J. Wu, and J. Yan. Statistical Methods and Computing for Big Data. *Stat. Interface*, 9(4):399–414, 2016a. ISSN 1938-7989. doi: 10.4310/SII.2016.v9.n4.a1. URL https://doi.org/10.4310/SII.2016.v9.n4.a1.

C. Wang, M.-H. Chen, E. Schifano, J. Wu, and J. Yan. Statistical Methods and Computing for Big Data. *Stat. Interface*, 9(4):399–414, 2016b. ISSN 1938-7989. doi: 10.4310/SII.2016.v9.n4.a1. URL https://doi.org/10.4310/SII.2016.v9.n4.a1.

M. Welling and Y. W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104568.

A. M. Yaglom. Certain Limit Theorems of the Theory of Branching Random Processes. *Doklady Akad. Nauk SSSR (N.S.)*, 56:795–798, 1947.

S. Zheng. Stochastic Approximation Algorithms in the Estimation of Quasi-Stationary Distribution of Finite and General State Space Markov Chains. *PhD Thesis, Columbia University*, 2014.

Part IV

Appendices

Appendix A

Quasi-Stationary Transition Density of Brownian Motion

Proof Let $\mathbf{X} = \{\mathbf{X}_s \,|\, 0 \leq s \leq t\}$ denote a given trajectory of a Brownian motion and ζ is the random variable associated with its potential killing time. For a given hazard rate κ, its survival probability in the interval $(s, s + ds)$ satisfies

$$\frac{\mathbb{P}(\zeta \in ds \,|\, \zeta > s, \mathbf{X})}{ds} \approx \kappa(\mathbf{X}_s), \tag{A.0.1}$$

which simplifies to

$$\frac{\mathbb{P}(\zeta \in ds \,|\, \mathbf{X})}{ds\, \mathbb{P}(\zeta > s \,|\, \mathbf{X})} = \frac{\mathbb{P}(\zeta > s \,|\, \mathbf{X}) - \mathbb{P}(\zeta > s + ds \,|\, \mathbf{X})}{ds\, \mathbb{P}(\zeta > s \,|\, \mathbf{X})} \approx \kappa(\mathbf{X}_s). \tag{A.0.2}$$

This takes the form of a differential equation as

$$\frac{d}{ds} \left(\log(\mathbb{P}(\zeta > s \,|\, \mathbf{X})) \right) \approx -\kappa(\mathbf{X}_s), \tag{A.0.3}$$

whose solution is

$$\mathbb{P}(\zeta > t \,|\, \mathbf{X}) \approx \exp\left(-\int_0^t \kappa(\mathbf{X}_s) ds \right). \tag{A.0.4}$$

Therefore, the unconditional probability for the survival of the process over all trajectories $\mathbf{X} = \{\mathbf{X}_s \,|\, 0 \leq s \leq t\}$ is the expectation over all such trajectories

between $\mathbf{X}_0$ and $\mathbf{X}_t$. This is given as the expectation over a Brownian bridge

$$\mathbb{P}(\zeta > t \,|\, \mathbf{X}_0, \mathbf{X}_t) \approx \mathbb{E}_{\mathbb{W}} \left(\exp\left(-\int_0^t \kappa(\mathbf{X}_s) ds \right) \middle| \, \mathbf{X}_0, \mathbf{X}_t \right), \tag{A.0.5}$$

where $\mathbb{E}_{\mathbb{W}}$ denotes the expectation with respect to the law of a Brownian bridge between $\mathbf{X}_0$ and $\mathbf{X}_t$. Letting $\mathbf{X}_0 = \mathbf{0}$ and $\mathbf{x}_t \in \mathbb{R}^d$, the transition density of a killed Brownian motion is

$$\mathbb{P}(\mathbf{X}_t \in d\mathbf{x}_t \,|\, \zeta > t) \propto \mathbb{P}(\mathbf{X}_t \in d\mathbf{x}_t) \mathbb{P}(\zeta > t \,|\, \mathbf{X}_t \in d\mathbf{x}_t) \tag{A.0.6}$$

$$\propto \exp\left(-\frac{||\mathbf{x}_t||^2}{2t} \right) \mathbb{E}_{\mathbb{W}} \left(\exp\left\{ -\int_0^t \kappa(\mathbf{X}_s) ds \right\} \middle| \mathbf{X}_0 = \mathbf{0}, \mathbf{X}_t = \mathbf{x}_t \right). \tag{A.0.7}$$

Appendix B

Calculations: Cauchy Example

B.1 Numerical Calculation of Rate of the Dominating Poisson Process

The drift function in the ReScaLE algorithm is given by

$$\frac{d}{dx}\log\left(\pi(x|\mathbf{y})\right) = \sum_{i=1}^{5}\left(\frac{2(y_i - x)}{1 + (y_i - x)^2} - \frac{1}{5}\frac{2x}{1 + x^2}\right). \tag{B.1.1}$$

We set the simulated data $\mathbf{y}$ in the global environment (as `data`) to circumvent its redefinition in each call. The source code for `drift` is given as the following:

```
#include <Rcpp.h>
#include <stdlib.h>
#include <vector>
using namespace Rcpp;
using namespace std;

static  std::vector<double> data{2.65226687,1.27648783,1.61011759,1.27433040,
0.08721209};
static int N = data.size();

double drift (double x){
  double result = 0;
  for (int i =0; i<N; i++){
    result = result + 2*(data[i] - x)/(1 + pow((data[i] - x),2));
  }
  result = result - 2*x/(1+x*x);
  return result;
}
```

The derivative of the drift function is

$$\frac{d^2}{dx^2}\log\left(\pi(x|\mathbf{y})\right) = \sum_{i=1}^{5}\left(\frac{-2(1-(y_i-x)^2)}{(1+(y_i-x)^2)^2} - \frac{1}{5}\frac{2(1-x^2)}{(1+x^2)^2}\right), \tag{B.1.2}$$

and its source code can be implemented as:

```
double divergence (double x){
  double result = 0;
  for (int i =0; i<N; i++){
    result = result - 2*(1-pow((data[i] - x),2))/pow((1 + pow((data[i] - x),2)),2);
  }
  result = result - 2*(1-x*x)/pow((1+x*x),2);
  return result;
}
```

We define $\phi(x)$ and carefully put the above source codes with the definition of

```
// [[Rcpp::plugins(cpp11)]]
// [[Rcpp::export]]
double phi(double x){
  double result = 0.5*(pow(drift(x),2) + divergence(x));
  return result;
}
```

in a file, `cauchy_toy_example.cpp`. We source function $\phi(\cdot)$ within our `R` environment by making a call to `Rcpp::sourceCpp('cauchy_toy_example.cpp')`. Finally, we make a call to `stats::optimise(f = phi, interval = c(-5,5))` which results into:

```
> stats::optimise(f = phi, interval = c(-5,5))
$minimum
[1] 1.249653

$objective
[1] -2.379829
```

Therefore, we set the value of $\Phi = -2.379829$ and define $\kappa(x) = \phi(x) - \Phi$. We add its source code

```
// [[Rcpp::plugins(cpp11)]]
// [[Rcpp::export]]
double kappa(double x){
  double result = phi(x)+2.379829;
  return result;
}
```

and make a call to itself using `Rcpp::sourceCpp('cauchy_toy_example.cpp')`. Finally, our aim is to find the rate of the dominating Poisson process K. Based on its behaviour in Figure (6.3), we achieve this numerically using `stats::optimise(f = kappa, interval = c(-5,0))`

```
> stats::optimise(f = kappa, interval = c(-5,0), maximum = TRUE)
$maximum
[1] -0.769506

$objective
[1] 13.99258
```

The above numerical optimization allows us to set the value of $K = 14$.

B.2 Numerical Integration of the Un-normalised Posterior Density

We write a source code of the un-normalised posterior density (6.2.2) for a vector of x values. This is given as

```
// [[Rcpp::plugins(cpp11)]]
// [[Rcpp::export]]
std::vector<double> unnormalised_posterior(std::vector<double> x){
  double prod;
  int Nx = x.size();
  std::vector<double> result(Nx);
  for (int j=0; j < Nx; j++){
    prod = 1;
    for(int i=0; i <N; i++){
      prod = prod*1/(1+pow((data[i]-x[j]),2));
    }
    prod = prod/(1+x[j]*x[j]);
    result[j] = prod;
  }
  return result;
```

```
}
```

We add this code to our original file and perform `Rcpp::sourceCpp()` for it to be accessed within our `R` environment. Finally, we perform a numerical integration using `R` command `stats::integrate`, and have

```
> stats::integrate(unnormalised_posterior, lower = -Inf, upper = Inf)
0.06281079 with absolute error < 9.6e-06
```

Appendix C

The Hazard Function for a Logistic Regression Model

We consider a general case where $\beta = (\beta_0, ..., \beta_{d-1}) \in \mathbb{R}^d$ denote a vector of coefficients in a logistic regression. If X denotes the design matrix, our target density is given by

$$\pi(\beta) \propto \prod_{i=1}^{n} p_i(\beta)^{y_i} \cdot (1 - p_i(\beta))^{1-y_i}, \tag{C.0.1}$$

where $p_i(\beta) = 1/(1 + \exp(-X_i\beta))$ and $\{y_1, ..., y_n\}$ are the binary observed values of the dependent variable. Here X_i is the i^{th} row of the design matrix where the first entry is 1. The gradient of the logarithm of $\pi(\beta)$ given by

$$\nabla \log(\pi(\beta)) = \nabla \left(\sum_{i=1}^{n} y_i \log(p_i(\beta)) + (1 - y_i) \log(1 - p_i(\beta)) \right). \tag{C.0.2}$$

We first note that

$$\frac{\partial}{\partial \beta_j} (p_i(\beta)) = \frac{1}{1 + \exp(-X_i\beta)} \times \frac{\exp(-X_i\beta)}{1 + \exp(-X_i\beta)} \times X_{i,j} \tag{C.0.3}$$

$$= p_i(\beta)(1 - p_i(\beta))X_{i,j}. \tag{C.0.4}$$

The gradient along the j^{th} dimension is given by

$$\nabla_{\beta_j} \log(\pi(\beta)) = \sum_{i=1}^{n} y_i \frac{1}{p_i(\beta)} \frac{\partial}{\partial \beta_j} (p_i(\beta)) - (1 - y_i) \frac{1}{(1 - p_i(\beta))} \frac{\partial}{\partial \beta_j} (p_i(\beta)) \tag{C.0.5}$$

$$= \sum_{i=1}^{n} y_i \frac{1}{p_i(\beta)} p_i(\beta)(1 - p_i(\beta)) X_{i,j} - (1 - y_i) \frac{1}{(1 - p_i(\beta))} p_i(\beta)(1 - p_i(\beta)) X_{i,j} \tag{C.0.6}$$

$$= \sum_{i=1}^{n} y_i (1 - p_i(\beta)) X_{i,j} - (1 - y_i) p_i(\beta) X_{i,j} \tag{C.0.7}$$

$$= \sum_{i=1}^{n} (y_i - p_i(\beta)) X_{i,j} \text{ for } j \in \{0, ..., d-1\}. \tag{C.0.8}$$

We should note that the divergence term is given by the sum of all diagonal entries of the Hessian matrix. In this context, the j^{th} diagonal term is

$$\nabla_{\beta_j^2} \log(\pi(\beta)) = \frac{\partial}{\partial \beta_j} \left(\nabla_{\beta_j} \log(\pi(\beta)) \right) \tag{C.0.9}$$

$$= \frac{\partial}{\partial \beta_j} \left(\sum_{i=1}^{n} (y_i - p_i(\beta)) X_{i,j} \right) \tag{C.0.10}$$

$$= -\sum_{i=1}^{n} p_i(\beta) (1 - p_i(\beta)) X_{i,j}^2 \text{ for } j \in \{0, ..., d-1\}. \tag{C.0.11}$$

Therefore, the $\phi(\cdot)$ takes the following form:

$$\phi(\beta) = \frac{1}{2} \left(\sum_{j=0}^{d-1} \left(\sum_{i=1}^{n} (y_i - p_i(\beta)) X_{i,j} \right)^2 - \sum_{j=0}^{d-1} \sum_{i=1}^{n} p_i(\beta) (1 - p_i(\beta)) X_{i,j}^2 \right). \tag{C.0.12}$$

In the above expression, the first term is always non-negative and

$$-p_i(\beta) (1 - p_i(\beta)) \geq -\frac{1}{4} \tag{C.0.13}$$

can be used to construct a lower bound Φ of $\phi(\cdot)$ as

$$\Phi = -\frac{1}{8} \left(\sum_{j=0}^{d-1} \sum_{i=1}^{n} X_{i,j}^2 \right). \tag{C.0.14}$$

Appendix D

Transformation: The Hazard Rate and an Unbiased Estimator

Here, we demonstrate a transformation of the hazard rate and establish the fact that under sub-sampling, we can achieve an $\mathcal{O}(1)$ rate for the dominating Poisson process within a given hypercube. In our case the posterior density is of the form

$$\pi(\mathbf{x}) \propto f_0(\mathbf{x}) \prod_{i=1}^{n} f_i(\mathbf{x}), \tag{D.0.1}$$

which contracts at a rate of $n^{-1/2}$. Therefore, we consider a transformation of our parameter $\mathbf{x}$

$$\mathbf{x} := \frac{1}{\sqrt{n}} (\Lambda_0 \mathbf{z}) + \hat{\mathbf{x}}, \tag{D.0.2}$$

where a matrix Λ_0 (often a diagonal matrix) is suitably chosen to scale the movement of each dimension of our killed Brownian motion. Here $\hat{\mathbf{x}}$ is a control variate in the space of $\mathbf{x}$ and $\mathbf{z}$ is the transformed variable which is a function of $\mathbf{x}$. As noted earlier, the hazard rate function is given by

$$\phi(\mathbf{x}) = \frac{\left|\left| \sum_{i=0}^{n} \nabla \log(f_i(\mathbf{x})) \right|\right|^2 + \sum_{i=0}^{n} \mathrm{Div} \nabla \log(f_i(\mathbf{x}))}{2}. \tag{D.0.3}$$

The gradient term under our transformation changes to

$$\nabla \log(f_i(\mathbf{x})) \longrightarrow \frac{1}{\sqrt{n}}\Lambda_0^\top \nabla \log\left(f_i\left(\frac{1}{\sqrt{n}}\left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}}\right)\right). \tag{D.0.4}$$

Next, we denote the second derivative matrix of $\log(f_i(\mathbf{x}))$ by $\nabla\nabla^\top \log(f_i(\mathbf{x}))$. Then our divergence term transforms to

$$\text{Div}\nabla \log(f_i(\mathbf{x})) \longrightarrow \frac{1}{n}\text{trace}\left(\Lambda_0^\top \left\{\nabla\nabla^\top \log\left(f_i\left(\frac{1}{\sqrt{n}}\left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}}\right)\right)\right\}\Lambda_0\right) \tag{D.0.5}$$

Therefore, the hazard rate under the transformation above can be rewritten as

$$\begin{aligned}\phi(\mathbf{z}) = \frac{1}{2n}\Bigg[\Bigg|\Bigg|\sum_{i=0}^{n}\Lambda_0^\top \nabla \log\left(f_i\left(\frac{1}{\sqrt{n}}\left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}}\right)\right)\Bigg|\Bigg|^2 + \\ \sum_{i=0}^{n}\text{trace}\left(\Lambda_0^\top \left\{\nabla\nabla^\top \log\left(f_i\left(\frac{1}{\sqrt{n}}\left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}}\right)\right)\right\}\Lambda_0\right)\Bigg].\end{aligned} \tag{D.0.6}$$

Next, we make the control variate adjustment within our hazard function. We first denote

$$\alpha(\mathbf{z}) := \sum_{i=0}^{n}\Lambda_0^\top \left\{\nabla \log\left(f_i\left(\frac{1}{\sqrt{n}}\left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}}\right)\right) - \nabla \log\left(f_i\left(\hat{\mathbf{x}}\right)\right)\right\}, \tag{D.0.7}$$

$$\alpha'(\mathbf{z}) := \sum_{i=0}^{n}\text{trace}\left(\Lambda_0^\top \left\{\nabla\nabla^\top \log\left(f_i\left(\frac{1}{\sqrt{n}}\left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}}\right)\right) - \nabla\nabla^\top \log\left(f_i\left(\hat{\mathbf{x}}\right)\right)\right\}\Lambda_0\right). \tag{D.0.8}$$

The hazard function can be then expressed as

$$\phi(\mathbf{z}) = \frac{1}{2n}\left[\left(\alpha(\mathbf{z})\right)\cdot\left[2\sum_{i=0}^{n}\Lambda_0^\top\left\{\nabla \log\left(f_i\left(\hat{\mathbf{x}}\right)\right)\right\} + \alpha(\mathbf{z})\right] + \alpha'(\mathbf{z})\right] + C, \text{ where} \tag{D.0.9}$$

$$\begin{aligned}C := \frac{1}{2n}\Bigg[\left(\sum_{i=0}^{n}\Lambda_0^\top\left\{\nabla \log\left(f_i\left(\hat{\mathbf{x}}\right)\right)\right\}\right)\cdot\left(\sum_{i=0}^{n}\Lambda_0^\top\left\{\nabla \log\left(f_i\left(\hat{\mathbf{x}}\right)\right)\right\}\right) + \\ \sum_{i=0}^{n}\text{trace}\left(\Lambda_0^\top\left\{\nabla\nabla^\top \log\left(f_i\left(\hat{\mathbf{x}}\right)\right)\right\}\Lambda_0\right)\Bigg].\end{aligned} \tag{D.0.10}$$

We can now find the lower bound Φ for $\phi(\mathbf{z})$ and construct $\kappa(\mathbf{z}) = \phi(\mathbf{z}) - \Phi$. We then run our ReScaLE algorithm in the space of $\mathbf{z}$. Finally, our samples in the $\mathbf{x}$

space can be found using

$$\mathbf{x} = \frac{1}{\sqrt{n}}\left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}}. \tag{D.0.11}$$

D.1 Sub-sampling of the Hazard Rate Under Transformation

As mentioned in Section (5.3), for $I, J \sim U\{0, ..., n\}$ we construct our unbiased estimator by first defining

$$\tilde{\alpha}_I(\mathbf{z}) := (n+1)\Lambda_0^\top \left\{ \nabla \log \left(f_I \left(\frac{1}{\sqrt{n}} \left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}} \right) \right) - \nabla \log \left(f_I \left(\hat{\mathbf{x}} \right) \right) \right\}, \tag{D.1.1}$$

$$\tilde{\alpha}'_I(\mathbf{z}) := (n+1)\text{trace} \left(\Lambda_0^\top \left\{ \nabla\nabla^\top \log \left(f_I \left(\frac{1}{\sqrt{n}} \left(\Lambda_0 \mathbf{z}\right) + \hat{\mathbf{x}} \right) \right) - \nabla\nabla^\top \log \left(f_I \left(\hat{\mathbf{x}} \right) \right) \right\} \Lambda_0 \right). \tag{D.1.2}$$

Now, our unbiased estimator can be constructed as

$$\tilde{\phi}\left(\mathbf{z}\right) := \frac{1}{2n} \left[\left(\tilde{\alpha}_I(\mathbf{z})\right) \cdot \left[2 \sum_{i=0}^{n} \Lambda_0^\top \left\{ \nabla \log \left(f_i \left(\hat{\mathbf{x}} \right) \right) \right\} + \tilde{\alpha}_J(\mathbf{z}) \right] + \tilde{\alpha}'_I(\mathbf{z}) \right] + C \tag{D.1.3}$$

Under the condition that the spectral radius of the second derivative matrix is bounded i.e. $\rho\left(\nabla\nabla^\top \log f_I(\mathbf{x})\right) \leq P_n$ for some $P_n > 0$, (ref to Section (5.3)) we have

$$|\tilde{\alpha}_I(\mathbf{z})| \leq (n+1) \cdot P_n ||\Lambda_0^\top \Lambda_0|| \frac{|\mathbf{z}|}{\sqrt{n}} \text{ and} \tag{D.1.4}$$

$$|\tilde{\alpha}'(\mathbf{z})| \leq (n+1) \cdot P_n \cdot d \cdot \text{trace} \left(\Lambda_0^\top \mathbf{1} \Lambda_0 \right). \tag{D.1.5}$$

Here d denotes the dimension of our state-space $\mathbf{z}$ and $\mathbf{1}$ is a $d \times d$ matrix with its all entries equal to 1. Therefore, the bound on ϕ can be calculated in the following way:

$$\begin{aligned} |\tilde{\phi}\left(\mathbf{z}\right) - C| \leq \frac{1}{2n} \Bigg[(n+1) \cdot P_n ||\Lambda_0^\top \Lambda_0|| \frac{|\mathbf{z}|}{\sqrt{n}} \Bigg[2 \left| \sum_{i=0}^{n} \Lambda_0^\top \left\{ \nabla \log \left(f_i \left(\hat{\mathbf{x}} \right) \right) \right\} \right| \\ + (n+1) \cdot P_n ||\Lambda_0^\top \Lambda_0|| \frac{|\mathbf{z}|}{\sqrt{n}} \Bigg] + (n+1) \cdot P_n \cdot d \cdot \text{trace} \left(\Lambda_0^\top \mathbf{1} \Lambda_0 \right) \Bigg]. \end{aligned} \tag{D.1.6}$$

We should note that the upper bound (D.1.6) is of $\mathcal{O}(1)$ which produces a iterative computational cost of $\mathcal{O}(1)$ within a QSMC method.

Appendix E

Rare Event Logistic Regression Code

The R code to generate data for a rare event logistic regression is given as:

```
# Set the number of observations to be simulated
n.obs <- 1000000

# Generate covariates using
mu <- c(0)
Sigma <- diag(1,1,1)
set.seed(1)
x <- MASS::mvrnorm(n.obs, mu, Sigma)

# Set actual values of coefficients and number of observations
beta <- c(-12.7, 1)

# Generate dependent variable as
set.seed(20)
y <- as.double(runif(n.obs) < 1 / (1 + exp(-beta[1] - x %*% beta[2])))
```

The above code results in a total of 8 successes, this is given as:

```
# The total number of successes is
> sum(y)
[1] 8
```

Appendix F

Code Availability

The `c++` and `R` code corresponding to examples considered in this thesis has been made available at https://github.com/DIVAKAR92/ReScaLE˙Examples.